




2014

Writing  
with Inform &  
The Inform  
Recipe Book




## Part I. Writing with Inform

Chapter 1: Welcome to Inform	Chapter 15: Numbers and Equations
Chapter 2: The Source Text	Chapter 16: Tables
Chapter 3: Things	Chapter 17: Understanding
Chapter 4: Kinds	Chapter 18: Activities
Chapter 5: Text	Chapter 19: Rulebooks
Chapter 6: Descriptions	Chapter 20: Advanced Text
Chapter 7: Basic Actions	Chapter 21: Lists
Chapter 8: Change	Chapter 22: Advanced Phrases
Chapter 9: Time	Chapter 23: Figures, Sounds and Files
Chapter 10: Scenes	Chapter 24: Testing and Debugging
Chapter 11: Phrases	Chapter 25: Releasing
Chapter 12: Advanced Actions	Chapter 26: Publishing
Chapter 13: Relations	Chapter 27: Extensions
Chapter 14: Adaptive Text and Responses	

-  Start reading here: §1.1. Preface
-  Part II. The Inform Recipe Book
-  Indexes of the examples and definitions

### Chapter 1: Welcome to Inform

*§1.1. Preface; §1.2. Acknowledgements; §1.3. The facing pages; §1.4. The Go! button; §1.5. The Replay button; §1.6. The Index and Results panels; §1.7. The Skein; §1.8. A short Skein tutorial; §1.9. Summary of the Skein and Transcript*

-  Contents of *Writing with Inform*
-  Chapter 2: The Source Text
-  Indexes of the examples

#### §1.1. Preface

Welcome to Inform, a design system for interactive fiction based on natural language.

Interactive fiction is a literary form which involves programming a computer so that it presents a reader with a text which can be explored. Inform aims to make the burden of learning to program such texts as light as possible. It is a tool for writers intrigued by computing, and computer programmers intrigued by writing. Perhaps these are not so very different pursuits, in their rewards and pleasures.

The sheer joy of making things... the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles... the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. (Frederick P. Brooks, "The Mythical Man-Month", 1972)

**Writing with Inform** is one of two interlinked books included with Inform: a concise but complete guide to the system. The other book is **The Inform Recipe Book**, a comprehensive collection of examples, showing its practical use. If you are reading this within the Inform application, you will see that the Writing with Inform pages are on "white paper", while the Recipe Book is on "yellow paper".


These notes are arranged so that the reader can, in principle, write whole works of fiction as early as the end of Chapter 3. Each subsequent chapter then extends the range of techniques available to make livelier and more intriguing situations.


This new release of Inform ("Inform 7", the seventh major version since 1993) is a radical departure from most previous approaches to interactive fiction. In particular, it is very different from Inform 6, which newcomers will not need to know anything about. Inform 6 sits inside Inform 7, and is part of the inner workings, but is not visible from the outside. For information about Inform 6, see [www.inform-fiction.org](http://www.inform-fiction.org).



Programming is best regarded as the process of creating works of literature, which are meant to be read... so we ought to address them to people, not to machines. (Donald Knuth, "Literate Programming", 1981)

★ See **Acknowledgements** for a chance to try out the cross-referencing links in *Writing with Inform* - click on the red asterisk or the name of the destination to go there

---

 Start of Chapter 1: Welcome to Inform

 Onward to §1.2. Acknowledgements

 Example 1:  **About the examples** An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

---

## §1.2. Acknowledgements

Inform 7 is dedicated to Emily Short and Andrew Plotkin, whose shrewd and sceptical suggestions made a contribution which can hardly be overstated. A long email correspondence with Andrew entirely subverted my original thoughts about natural-language IF, as he convinced me that the "new model" of rule-based IF was a truer foundation; while Emily's wry, witty analysis and how-about-this? cheered me at low moments, besides providing the impetus and often the specifics for a lot of the best ideas.

From the outset, I have thought of Inform 7 as no longer being a command-line compiler, but a compiler in combination with a humanising user interface. All credit for the reference implementation under Mac OS X belongs to Andrew Hunter. How simple the metaphor of an interactive book with facing pages may seem, but the coding was an enormous challenge. In

2014 Toby Nelson, the author's brother, put months of time into the project by rewriting and modernising the Mac OS X application: sandboxing it for the Mac App Store, giving it a more contemporary design, and much more.

Though David Kinder's Windows application does indeed visually follow the OS X original, the two programs were coded independently, and the programming task taken up by David was formidable indeed. Philip Chimento's Gnome-based user interface for Linux became officially part of the project in November 2007, when the first easy-to-install packages for Ubuntu and Fedora were offered. Philip's efforts were particularly generous since the early stages of Inform-for-Linux were so tentative: for many months, we weren't sure how to go about the project, and during that time Philip quietly wrote us a solution. Adam Thornton continues to support Inform at the command line on Unix-like systems.


Inform in its widest sense incorporates work by so many people that it's simply impossible to thank all of them, but Erik Temple, Dannii Willis, Ron Newcomb, Eric Eve and Juhana Leinonen all deserve special mention. More than 250 users have filed patient and careful bug reports, keeping us on the straight and narrow. They're contributors, too.


It's perhaps surprising that the Inform application gained its first online component only in 2014, but the Public Library of Extensions, and its discussion forum, make a brave step into the modern age. Mark Musante, our Extensions librarian, Dannii Willis and Justin de Vesine have been a great help in setting this up.


The original development of Inform 7 was a long haul, and I would particularly like to thank Sonja Kesserich, David Cornelson and other volunteers for their early testing of a then-fragile system. The final months before the Public Beta release of Inform 7 were made more enjoyable, as well as more productive, by fruitful discussions leading to a cross-platform standard for bibliographic data and cover art. L. Ross Raszewski, who wrote frighteningly efficient reference software in frighteningly little time; the librarians of the IF-Archive, Andrew Plotkin, David Kinder and Paul Mazaitis; and my fellow authors of IF design systems - Mike Roberts (of the Text Adventure Development System); Kent Tessman (of Hugo); and Campbell Wild (of ADRIFT).

This EPUB edition of the documentation was greatly assisted by excellent advice published by Liza Daly, an old friend of Inform's who also helped construct our website.

---

 Start of Chapter 1: Welcome to Inform

 Back to §1.1. Preface

 Onward to §1.3. The facing pages

---




### §1.3. The facing pages

On most computers, Inform runs in a single main window which is an opened book showing two facing pages. As we shall see it behaves as if these pages are in dialogue with each other: for the most part we write on the left hand page and see responses appear on the right. But all is controllable. The margin between the two pages can be dragged back and forth like the slide on a trombone: each page can be made smaller that the other may grow larger. Moreover, each page can display one of a number of displays relevant to the current project,

called "panels", one of them being the Documentation panel which displays a screen-readable copy of this manual. The vertical strip of choices at the right hand margin of each page allows you to choose between panels. (The same panel can be showing on both pages at the same time, if that's useful.)

At the start the only panels available are a blank space in which to write the first lines of a new interactive fiction - the Source panel - and this one, the Documentation. Clicking on the other choices will do nothing.

The exception is the Settings panel, which contains some preference settings for the individual project - not the whole application. This is always available, but it controls settings which can be left alone almost all of the time.

- 
-  Start of Chapter 1: Welcome to Inform
  -  Back to §1.2. Acknowledgements
  -  Onward to §1.4. The Go! button
- 

## §1.4. The Go! button

Clicking the Go button translates the text in the Source panel into a computer program which enacts the interactive fiction, and automatically sets it going (in the Story panel, which opens as needed).

If the Source is empty of text, Inform will be unable to create anything: it needs at least one name of a location where the drama can unfold. For reasons of tradition, such locations are normally called "rooms", though people have used them to represent anything from grassy fields to states of mind and other metaphorical places.




"Midsummer Day"

The Gazebo is a room.

Clicking Go with this text in the Source panel will result in a short delay, after which the Story panel will appear, from which we can explore this newly created world: an interactive fiction called "Midsummer Day". It will not be very exciting, since Inform has only five words to go on, but we can add more detail to the source at any point and then click Go again to try out the changes. (Note that there is no need to "quit" these explorations in the Story panel. When Go is clicked, any story already in progress is discarded in favour of the new version.)

The keyboard shortcut Command-R (on Mac OS X), F5 (on Windows), or Ctrl-R (on Linux GNOME) has the same effect as clicking Go.

---

-  Start of Chapter 1: Welcome to Inform
  -  Back to §1.3. The facing pages
  -  Onward to §1.5. The Replay button
- 

## §1.5. The Replay button

Replay works identically to Go, except that it does something further: once the story is created, it automatically plays through the same commands as were typed into the previous version. For instance: suppose we click Go to bring Midsummer Day into being, and find ourselves playing the story. We type "look" and find that there is not much to see. Going back to the source, we add

"A white canvas parasol raised up on stakes driven into the grass."




so that the source now reads

"Midsummer Day"

The Gazebo is a room. "A white canvas parasol raised up on stakes driven into the grass."

Instead of clicking Go, we click Replay, and can sit back and watch what has changed. In this example, it only saves us the trouble of typing "look", but once stories become long and elaborate, Replay is invaluable: and especially when we notice in play that something very minor is wrong - a spelling error, say - and want to fix it immediately, without fuss.


---


-  Start of Chapter 1: Welcome to Inform
  -  Back to §1.4. The Go! button
  -  Onward to §1.6. The Index and Results panels
- 


## §1.6. The Index and Results panels

If, when Go! is clicked, the text in the Source panel is not fully understood, then Inform will generate a report of the problems it found, which will open in the "Errors" panel. (Other information is also available in "Errors", but most of it is used for debugging Inform, and can be ignored.)


On the other hand, if the text was fully understood then another new panel will become available: the "Index". This is a cross-referenced index of the source, or rather, of the interactive fiction which has been generated. The Index is only an optional convenience, but becomes more and more helpful as the fiction grows larger. Its exact format does not matter for now.


The icon  always denotes a reference to a particular line in the Source text, that is, to something written in the source: clicking it opens the Source panel and jumps to that position.


The icon  indicates that more detailed information can be read further down the text in the same panel: clicking it jumps down to this more detailed report.

Lastly, the icon  hints that there is a relevant page of this manual: clicking this opens the Documentation panel and switches to it.

---

 Start of Chapter 1: Welcome to Inform

 Back to §1.5. The Replay button

 Onward to §1.7. The Skein

---

## §1.7. The Skein


The Replay button demonstrates that Inform must be quietly remembering the commands typed into the last run through the story. In fact it remembers, and automatically organises, *every* previous run.


Inform's approach to testing interactive fiction is to treat it as being like the analysis of other turn-based games, such as chess. It would be prohibitively difficult to work out every possible combination of moves: instead, we analyse those which go somewhere, and look for significant choices. Every Queen's Gambit begins with the same first three moves (1. d4, d5; 2. c4), but then there is a choice, as the next move decides whether we have a Queen's Gambit Accepted (dxc4) or Declined (e6). Books about chess often contain great tables of such openings, which run together for a while but eventually diverge. To learn chess, one must explore all of these variations.


Inform's Skein panel is just such a table, built automatically. If we think of the list of typed commands as a thread, then the skein is (as the name suggests) braided together from all these threads. In the display, time begins at the top, with the **start** knot, and the threads of different play-throughs hang downwards from it.

Double-clicking on a command translates the source afresh and replays the story from **start** down to that command, and then stops. We are then free to continue play by typing commands into the Story panel, of course, and these commands will automatically be recorded in the Skein as a new variation of play, diverging from the previous threads.

---

 Start of Chapter 1: Welcome to Inform

 Back to §1.6. The Index and Results panels

 Onward to §1.8. A short Skein tutorial

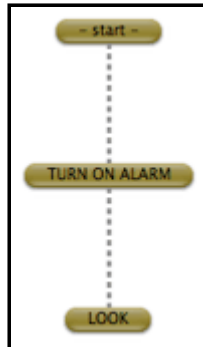
---

## §1.8. A short Skein tutorial

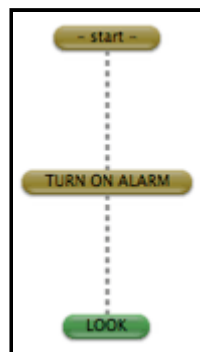
In the following example, we will see how the Skein is woven as different commands are tried. As it happens, the story being played is the example "Witnessed", from Chapter 11, but the details do not matter. When the project has never been played at all, if we switch to the Skein panel (or open it opposite the Story panel) we will only find this:



Suppose we click Go for the first time and type two commands in: TURN ON ALARM and then LOOK. Now the Skein shows:

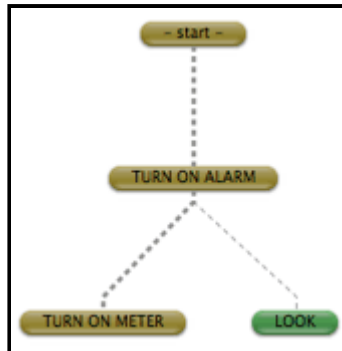


Only one line of play is known to Inform, and it runs downwards in a thread from the special "- start -" knot, which represents the situation before any command has been tried. The useful thing about having past histories recorded like this is that we can revisit them. Suppose we want to go back to the situation after typing only TURN ON ALARM. We could click Go again and type that first command in once more, but now we have an easier method: we simply double-click on the TURN ON ALARM knot. The story restarts by itself, and commands are automatically keyed in to regain the position of play represented by the knot we clicked on - in this example that only keys a single command in, but it might have been hundreds. The Skein now looks like this:



All knots are displayed either as yellow or green. Yellow knots are the ones in the history of the story currently playing. The LOOK knot is green because it hasn't happened in the current story yet - and in fact, it won't happen in the current story, because instead we play TURN ON METER. Now the Skein changes again:





Inform now knows about two ways to play the current project: one consisting of TURN ON ALARM and then TURN ON METER, the other of TURN ON ALARM and then LOOK. Since these only differ after the first turn, Inform displays them as a thread which divides into two after the first turn. Again, LOOK remains green because it hasn't been played in the current story.

Note also that one of the two possible threads here is drawn more thickly (here it is shown with thick dashes rather than thin). Only one thread is ever drawn thickly -- the one currently being shown in the Transcript panel, which we will come to later on. (That often corresponds to the current line of play, as now, because the Transcript follows what we do unless we choose otherwise.)

After a little more exploration, we reach the following:



At this point we decide that we want to preserve the thread leading to EXAMINE CHIMES - perhaps it's a sequence we are going to want to test often. The Skein can be edited very easily: right-clicking on a knot brings up a contextual menu.

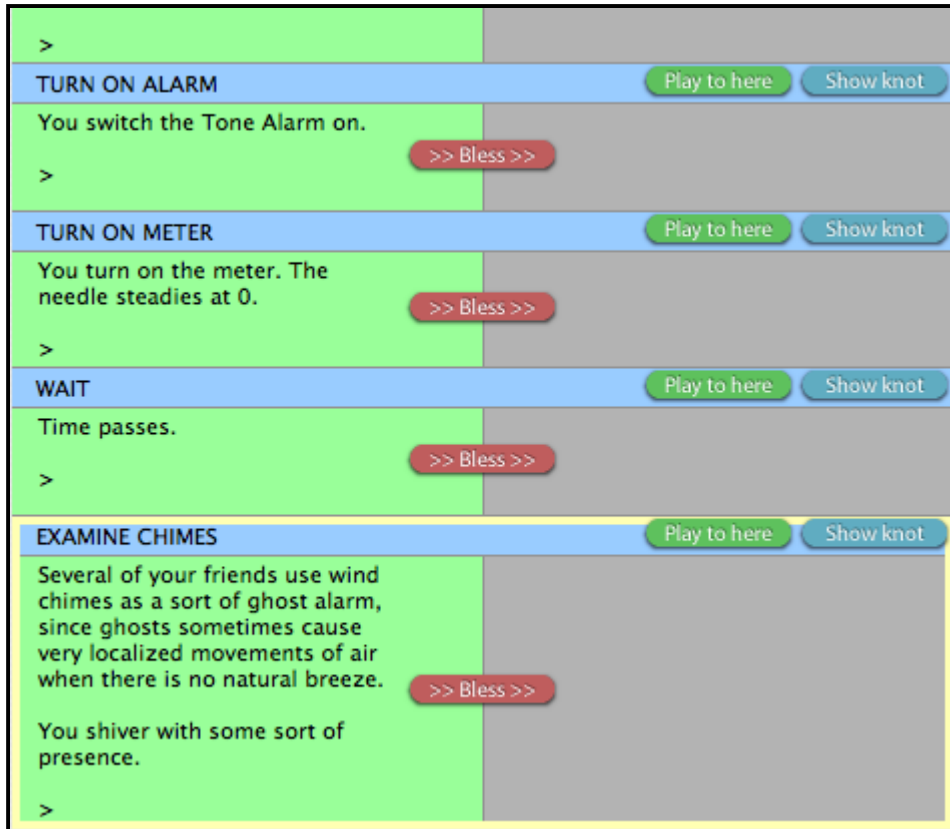
We choose Lock This Thread from the contextual menu, and this makes the thread through to here "locked". That means the knots can't be deleted (unless we unlock them again) - either by our own mistake, or by Inform trimming back no-longer-needed threads of the Skein to keep it manageable in size.



Note that this locked history is now drawn as a solid thread, whereas all the others are unlocked and drawn as dashes.

Now we have a securely remembered piece of standard play: it means we can try out the sequence TURN ON ALARM / TURN ON METER / WAIT / EXAMINE CHIMES any time we want to with a double-click on the final knot. This is convenient for testing - but so far it only runs the test: to see whether the test came out well or badly, we have to look through what happened, perhaps by scrolling back in the Story panel to look at the text. And that means that we need to remember what the text should have been like.

In fact, though, Inform can remember for us, using the Transcript panel. This is closely joined to the Skein panel, and it's often convenient to flip between the two. Turning to the Transcript now, we find a two-column view of the story currently being played. The left-hand column shows the text which has been displayed on each turn so far; the right-hand column is empty. The bottom of the Transcript looks like so:



The empty right-hand column displays the "blessed" transcript - one which the author has approved as being correct. This can be done for each individual knot, using the Bless button joining the columns, but in this case we will bless the whole transcript of this story, using the Bless All button. Now there's text in both columns, and of course the two columns match. (Note that the blessed transcript is in a brighter colour.)

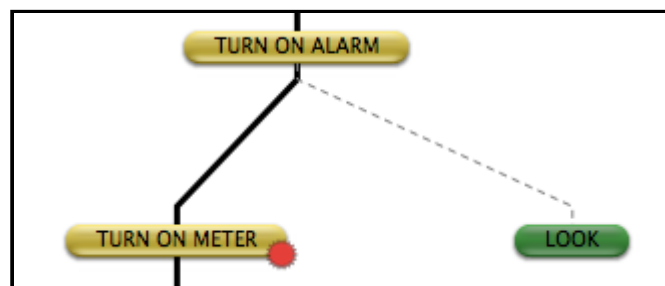


Back in the Skein, we find that the knots which have transcripts have lit up, and are brighter than the others. If we Go, to start a new story, and then look at the Skein:

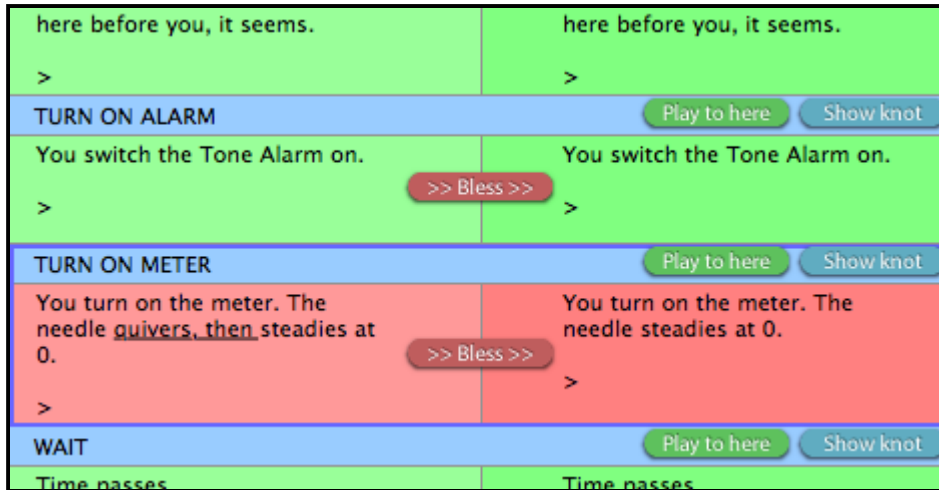


we see that the knots for which we have blessed a transcript are in a brighter green (or a brighter yellow, if they're in the current story being played).

Now suppose we change the source text for the project, so that we make it behave differently. The details don't matter, but suppose we do something which changes the result of the TURN ON METER command, and then run the test again. Now we find:



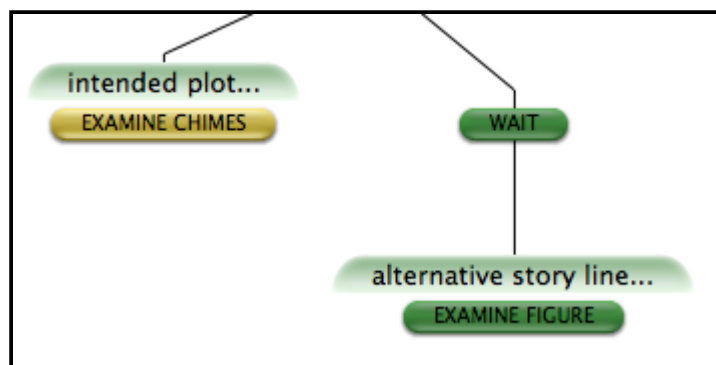
The red warning badge on the TURN ON METER knot alerts us that the last time this knot was tried (just now, as it happens), the resulting text didn't agree with its blessed transcript. (Red badges can only be seen on bright-coloured knots which have transcripts - for other knots, there's nothing to compare with.) On the other hand, the rest of the yellow current line of play worked out exactly the same as we expected - so no badges. Clicking on the red badge takes us into the Transcript panel at the right place, where the corresponding turn's transcript has also turned red:



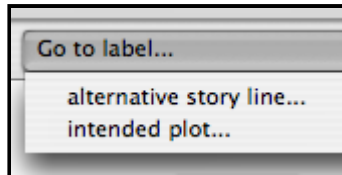
Again, what actually happened is on the left; what should have happened is on the right. The change is shown with underlining - we added the text "quivers, then". If we approve this change, by clicking on the Bless button for the red turn, the amended text will become the correct text to compare against in future runs, and the turn will become green to show that once again all is well. (We can also edit the blessed transcript directly, by double-clicking in the text and typing.) Clicking on the Show knot button takes us back in the skein, at the right place: where we will see that the red warning badge has disappeared.

Some writers of IF like to work backwards from a transcript of the story they want to produce, and for them, the Skein and Transcript combination will be helpful as a running picture of what works so far. Other authors may not use the Skein/Transcript feature at all until right at the end of a project, in testing before publication, when it becomes very important to be able to make small changes in one area without upsetting everything else. Either way, the Skein and Transcript together make a very powerful testing aid.

This tutorial has shown only a short line of play, to keep the pictures small, but for a large project the Skein might run to thousands of knots. It then becomes important to be able quickly to find key knots corresponding to plot developments. To help with that, we can annotate certain knots with any label we choose (by selecting Add Label from the contextual menu):






And this is where the "Labels" gadget at the top of the Skein comes into its own:



since it offers a menu of all the labels in the Skein, and if selected will jump to the one chosen.

The Skein has other abilities too, best explored by experimenting. For instance, we can edit the commands by selecting Edit Knot from the contextual menu, and we can add new knots in the middle of existing lines using the Insert Knot item on that menu. The Play All Blessed option (on the Game menu) is especially powerful: it tests each possible blessed history in turn, trying all of them, and can therefore test very complicated multiple endings and the like in a single click.

- 
-  Start of Chapter 1: Welcome to Inform
  -  Back to §1.7. The Skein
  -  Onward to §1.9. Summary of the Skein and Transcript
- 

## §1.9. Summary of the Skein and Transcript

The Skein records the history of different plays through the current project, and the Transcript records the text of each response, comparing it with a "blessed" or correct version if one is available.




In the Skein each typed command is a "knot". The threads hanging down from the top "start" knot are possible histories. Double-click on a knot to play through to there.

Yellow knots are commands played so far in the current story: green knots are possible lines not taken, or not taken yet.

A solid thread is "locked" and protected from deletion (by accident or when Inform trims away loose ends): a dashed thread has no such protection.

A bright knot has a blessed transcript: a darker knot is one which has no blessed transcript. When a bright knot shows a red badge, this means that when last tested its command produced a textual reply which wasn't the same as the blessed transcript. Clicking on the badge shows exactly how.

The thicker thread in the Skein shows the history currently being displayed in the Transcript panel.

- 
-  Start of Chapter 1: Welcome to Inform
  -  Back to §1.8. A short Skein tutorial
  -  Onward to Chapter 2: The Source Text: §2.1. Creating the world
-

## Examples from Chapter 1: Welcome to Inform



Start of this chapter



Chapter 2: The Source Text



Indexes of the examples

1


### ★ Example About the examples

An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

RB

This is the first of about 400 numbered examples. In a few cases, such as this one, they provide a little background information, but almost all demonstrate Inform source text. The techniques demonstrated tend to be included either because they are frequently asked for, or because they show how to achieve some interesting effect.

The same examples are included in **both** of the books of documentation, but in a different order: in *Writing with Inform*, they appear near the techniques used to make them work; in *The Inform Recipe Book*, they are grouped by the effects they provide. For instance, an example called "Do Pass Go", about the throwing of a pair of dice, appears in the "Randomness" section of *Writing with Inform* and also in the "Dice and Playing Cards" section of *The Inform Recipe Book*. Clicking the italicised WI and RB buttons at the right-hand side of an example's heading switches between its position in each book.

Many computing books quote excerpts from programs, but readers have grown wary of them: they are tiresome to type in, and may only be fragments, or may not ever have been tested. The authors of Inform have tried to avoid this. All but two dozen examples contain entire source texts. A single click on the paste icon  (always placed just left of the double-quoted title) will write the complete source text into the Source panel. All that is then required is to click the Go button, and the example should translate into a working game.

In most cases, typing the single command TEST ME will play through a few moves to show off the effect being demonstrated. (You may find it convenient to create a "scratch" project file for temporary trials like this, clearing all its text and starting again with each new test.)

As part of the testing process which verifies a new build of Inform, each example in turn is extracted from this documentation, translated, played through, and the resulting transcript mechanically checked. So the examples may even work as claimed. But the flesh is weak, and there are bound to be glitches. We would welcome reports, so that future editions can be corrected.

Each example is loosely graded by difficulty: if they were exercises in a textbook, the asterisks would indicate how many marks each question scores. As a general rule:

★ - A simple example, fairly easily guessed.

★★ - A complicated or surprising example.

★★★ - An example needing detailed knowledge of many aspects of the system.

★★★★ - A complete scenario, containing material not necessarily relevant to the topic being demonstrated.





In general, the main text of *Writing with Inform* tries never to assume knowledge of material which has not yet appeared, but the trickier examples almost always need to break this rule.

---



## Chapter 2: The Source Text

§2.1. *Creating the world*; §2.2. *Making rules*; §2.3. *Punctuation*; §2.4. *Problems*;  
§2.5. *Headings*; §2.6. *Why using headings is a good idea*; §2.7. *The SHOWME command*;  
§2.8. *The TEST command*; §2.9. *Material not for release*; §2.10. *Installing extensions*;  
§2.11. *Including extensions*; §2.12. *Use options*; §2.13. *Administering classroom use*;  
§2.14. *Limits and the Settings panel*; §2.15. *What to do about a bug*;  
§2.16. *Does Inform really understand English?*

-  Contents of *Writing with Inform*
-  Chapter 1: Welcome to Inform
-  Chapter 3: Things
-  Indexes of the examples

### §2.1. Creating the world

Designing an interactive fiction can be divided into two related activities. One is the creation of the world as it appears at the start of play: where and what everything is. The other is to specify the rules of play, which shape how the player interacts with that initially created world. A new Inform project is void and without form, so to speak, with nothing created: but it starts with hundreds of standard rules already in place.

The same division between creating things, and laying down rules, is visible in Inform source text. The creation of the world is done by making unconditional factual statements about it. For example,

*The wood-slatted crate is in the Gazebo. The crate is a container.*

Inform calls sentences like these "assertions". The verb is always written in the present tense (thus the crate "is", not "will be"). Further examples are:

*Mr Jones wears a top hat. The crate contains a croquet mallet.*

The words "is", "wears" and "contains" are forms of three of the basic verbs built in to Inform. There are only a few built-in assertion verbs, of which the most important are *to be*, *to have*, *to carry*, *to wear*, *to contain* and *to support*. (As we shall see, further assertion verbs can be created if needed.)

The world described by these assertions is the starting condition of the story: what happens when play begins is another matter. If somebody picks up the crate and walks off with it, then it will no longer be in the Gazebo. Mr Jones may remove his hat.

---



Start of Chapter 2: The Source Text



Back to Chapter 1: Welcome to Inform: §1.9. Summary of the Skein and Transcript



Onward to §2.2. Making rules

---

## §2.2. Making rules

The other kind of sentence tells Inform what should happen in certain circumstances, and reads like an instruction issued to someone:

Instead of taking the crate, say "It's far too heavy to lift."

This is a "rule", and it changes the crate's behaviour. The player who tries typing "take crate", "pick up the crate" or similar will be met only with the unhelpful reply "It's far too heavy to lift." The many different kinds of thing which the player can do are called "actions", and are always written as participles: "taking ...", for instance, or "putting ... on ...".

Inform is built on a mass of several hundred rules, some quite complex, and it could even be said that Inform *is* that mass of rules. We never see the complexity behind the scenes because the whole aim is to provide a basic, penny-plain, vanilla flavoured sort of realism. It would be surprising if one could put the crate inside itself, so a rule exists to forbid this. It would be surprising if one could drop something which was already on the ground, and so on. These basic rules of realism are the ones which every new Inform project starts with.

A rule always starts with a situation which it applies to, and then follows with one or more things to do. Here's an example where the situation is "Before taking the crate" - the player is just starting to try to pick the box up - and there's a three-step process to follow, but steps 2 and 3 happen only if step 1 comes out in a particular way:

Before taking the crate:

if the player is wearing the hat:

now the hat is in the crate;

say "As you stoop down, your hat falls into the crate."

The steps to follow here are called "phrases". Inform knows about 400 built-in phrases, but most of them are needed only occasionally. These three are used over and over again:

**if** tells Inform to do something only if some "condition" holds, here "the player is wearing the hat";

**now** tells Inform to change the situation, here so that the hat moves to the crate; and

**say** tells Inform to say something, that is, to write some text for the player to read.

Every one of the built-in phrases has a definition somewhere in this book. The full definition of "say" will come later, but in the simple form above it writes out the given text for the player to read. (Normally this text is simply shown on screen, not spoken aloud, unless software adapted for partially sighted people is being used.) Phrase definitions are all linked to in the Phrases page of a project's Index.

---

↑ Start of Chapter 2: The Source Text

← Back to §2.1. Creating the world

→ Onward to §2.3. Punctuation

---

## §2.3. Punctuation

An example rule from the previous section demonstrates one of Inform's conventions about punctuation, and is worth pausing to look at again.

Instead of taking the crate, say "It's far too heavy to lift."

In English grammar, it's usual to regard a full stop as closing its sentence even when it occurs inside quotation marks, provided there is no indication to the contrary, and this is also the rule used by Inform. Thus:

The description is "Shiny." It is valuable.

is read as equivalent to

The description is "Shiny.". It is valuable.

Sentence breaks like this occur only when the final character of the quoted text is a full stop, question mark or exclamation mark (or one of these three followed by a close bracket) and the next word begins, in the source code, with a capital letter. A paragraph break also divides sentences, behaving as if it were a full stop.

Material in square brackets [like so] is "comment", in computing jargon: it is considered as being an aside, a private note by the author, and not read in by Inform. This allows us to make notes to ourselves like so:

The China Shop is a room. [Remember to work out what happens if the bull gets in here!]

Inform is all about text, so pieces of text are often quoted in Inform source. This example is typical:

The description is "Shiny." It is valuable.

Quotations always use double-quotation marks, which aren't part of the text. So the description here is just the five letters and full stop in between the marks:

Shiny.

That seems straightforward, but there are three conventions to watch out for.

1. Square brackets [ and ] inside quoted text don't literally mean [ and ]. They're used to describe what Inform should say, but in a non-literal way. For example,

"Your watch reads [time of day]."

might produce

Your watch reads 9:02 AM.

These are called "text substitutions". They're highly flexible, and they can take many different forms.

2. Single quotation marks at the edges of words are printed as double. So:

"Simon says, 'It's far too heavy to lift.'"

produces

Simon says, "It's far too heavy to lift."

3. Texts which end with sentence-ending punctuation - full stop, question mark, exclamation mark - are printed with a line break after them. So:

say "i don't know how this ends";  
say "I know just how this ends!";

would come out quite differently - this doesn't affect the appearance of the text, but only the position where the next text will appear. Something to be careful about is that this only applies when the punctuation occurs at the end of a "say", as in these examples. (It doesn't apply when a varying textual value is printed, using some text substitution, because then the pattern of where line breaks occur would be unpredictable - sometimes the value might end in a punctuation mark, sometimes not.)

These three punctuation rules for texts feel very natural with practice, and Inform users sometimes don't realise the third rule is even there, because it just seems the right thing to happen. But occasionally the rules get in the way of what we want to do. (For instance, how do we get a literal [ or ]? What if we want a single quote mark where Inform thinks we want a double, or vice versa?) So we'll come back to these rules in more detail in the chapter on Text.


Inform also reads other punctuation marks. Colon ":" and semicolon ";" turned up in the previous section, in the writing of rules. It also has the more exotic "|" (not a capital I, a vertical stroke) for paragraph breaks outside of quoted text, but people hardly ever need this.

As these examples begin to show, Inform source imitates the conventions of printed books and newspapers whenever there is a question of how to write something not easily fitting into words. The first example of this is how Inform handles headings, but to see why these are so useful we first look at Problems.

★ See **How Inform reads quoted text** for a fuller exploration of the punctuation rules for text

---

 Start of Chapter 2: The Source Text



 Back to §2.2. Making rules

 Onward to §2.4. Problems



---

## §2.4. Problems

The language used in the source reads as if it were English aimed at a human reader (and this is intentional: the designer, after all, is a human reader and needs to be able to understand his or her own source), but in reality Inform can only understand a very modest range of sentences and will complain if its limits are passed. Subtler problems arise if the source contains contradictions. For instance, the following "Problem" might be produced:


**Problem.** You wrote 'A starting pistol is in the cup' , but in another sentence 'A Panama hat is on the cup' : the trophy cup cannot both contain things and support things, which is what you're implying here. If you need both, the easiest way is to make it either a supporter with a container attached or vice versa. For instance: 'A desk is here. On the desk is a newspaper. An openable container called the drawer is part of the desk. In the drawer is a stapler.'

This is a rather discursive error message, and if a similar problem were to occur in the same run through, it would be curtailed to:

**Problem.** You wrote 'A firing pistol is in the box' , but in another sentence 'A fedora hat is on the box' : again, the croquet box cannot both contain things and support things.

---

 Start of Chapter 2: The Source Text

 Back to §2.3. Punctuation

 Onward to §2.5. Headings

---

## §2.5. Headings

Once the source grows beyond 1000 words or so, it can all too easily become disorganised, and by the time it reaches the size of a novella it can be difficult to find things. (Though nearly all editors provide a Find function, and the keyboard short-cuts Command-F (Mac OS X) or Ctrl-F (Windows) do the trick in the Inform application.)

Inform provides for us to organise the source code in just the way that a printed book would be organised: with headings and subheadings. Firstly, we can put the title at the top. If the first paragraph consists only of a single quoted piece of text, then that's the title; and an author can also be given, as follows:

"Spellbreaker" by Dave Lebling

We will later see that more bibliographic information can also be placed here, in the same way that the imprint page of a novel comes before the text gets going. The author's name can

normally be given without quotation marks, so long as it contains no punctuation. For instance:




"Three Men in a Boat" by "Jerome K. Jerome"

needs quotes as otherwise the full stop after the K will be mistaken for the end of a sentence.


A sentence which is the only one in its paragraph and which begins with any of the words "volume", "book", "part", "chapter" or "section" is considered to be a heading or a sub-heading. It must not contain a typed line break, and in order to stand alone in its paragraph there should be a skipped line both before and after it. For instance:

Section 2 - Flamsteed's Balloon



Headings can be written in any format, provided they start with one of the five indicator words, and they are hierarchical: a "Part ..." heading is considered more significant than a "Chapter ..." heading but not so significant as a "Book ..." heading, and so on. (We do not need to use all five kinds of heading.)

- 
-  Start of Chapter 2: The Source Text
  -  Back to §2.4. Problems
  -  Onward to §2.6. Why using headings is a good idea
- 



## §2.6. Why using headings is a good idea

Reports of problems, as we have seen, often quote back the source to justify themselves. Rather than quoting line numbers ("Midsummer Day, line 2017" or something similar) Inform uses the  icon. The down side of this is that a glance at the list of problems might give little hint of whereabouts in the source the difficulties lie. Inform therefore makes use of headings to give a general indication:

In Part the First, Chapter 1 - Attic Area:

**Problem.** You wrote 'South of the Attic is the Winery' , but in another sentence 'South of the Attic is the Old Furniture' : this looks like a contradiction, which might be because I have misunderstood what was meant to be the subject of one or both of those sentences.

In Chapter 2 - Deeper In:




**Problem.** You wrote 'The Disused Observatory is south of the Dark Room' , but in another sentence 'South of the Dark Room is the Cupboard' : again, this looks like a contradiction.

Secondly, headings are used in the Contents page of the Index, and they allow rapid navigation through the source, by jumping to any heading or subheading with a single click.

Finally, headings are used when working out what a name refers to. Suppose the source contains both a "four-poster bed" and also a "camp bed", and we write something like "The pillow is on the bed." Inform decides which bed is meant by giving priority to whichever is defined in the current section (so far), or failing that the current chapter, or current part, or current book, or finally the current volume. This allows us to write, for instance,

The four-poster bed is in the Boudoir. The pillow is on the bed.

and not have the pillow mysteriously turn up on the camp bed, which hasn't been mentioned since way back in Chapter 2.

- 
-  Start of Chapter 2: The Source Text
  -  Back to §2.5. Headings
  -  Onward to §2.7. The SHOWME command
- 

## §2.7. The SHOWME command

Problem messages are generated when the source text does not make sense to Inform. Even if it does make sense, though, there is no guarantee that it does what the author intends, and the only way to find out is to test the result by playing through it (or asking others to). For the most part one plays as if one were the eventual reader of the work, but sometimes it is highly convenient to have the god-like powers which are an author's prerogative. These are provided by the testing commands, which are present at every stage until the final release version (generated by the Release button). They will be introduced in this manual as they become relevant: here is the first.

The testing command SHOWME prints out a brief summary about a room or thing, and any contents or parts it may have. Typing SHOWME on its own shows the current room, but any item or room in the story, however distant, can be named instead. For instance:




```
>showme
Boudoir - room
  four-poster bed - supporter
  yourself - person
  pillow
```

```
>showme diamonds
diamonds - thing
location: in the strongbox on the dresser in the Drawing Room
unlit; inedible; opaque; portable; singular-named; improper-named
description: The diamonds glitter dangerously.
printed name: diamonds
```

Much of this can be seen, and seen more easily, in the World tab of the Index panel: but that only shows the initial state of play, whereas the SHOWME command reveals the situation in mid-story. ("Room", "supporter" and so on are kinds, of which more in Chapter 3.)

★ See **High-level debugging commands** for more convenient testing commands like this one

---

-  Start of Chapter 2: The Source Text
  -  Back to §2.6. Why using headings is a good idea
  -  Onward to §2.8. The TEST command
- 

## §2.8. The TEST command

The only way to thoroughly test a work of IF is to run a complete solution through it, and carefully check the resulting transcript of dialogue. The Skein and Transcript tools of the Inform application are provided for exactly this purpose.

All the same, most works of interactive fiction contain occasional vignettes, either in terms of short scenes of narrative, or in the behaviour of particular things or rooms, which we would like to test without the fuss of using the full story-level Skein tool. The examples in the documentation are like this: in almost every example, typing TEST ME puts the story through its paces.

Solutions or sequences for testing ("scripts") can be defined with sentences like so:

`Test balloon with "get balloon / blow balloon / drop balloon".`

This has no effect on the design itself, but ensures that when the story is played, typing "test balloon" will run through the given three commands in sequence, as if we had typed "get balloon" and then "blow balloon" and then "drop balloon".

The name for the test (balloon in this example) has to be a single word. Typing just "test" at the story prompt gives a list of all the test scripts known to the story. Test scripts can make use of each other, for instance:

`Test all with "test balloon / test door".`

One convenient way to keep track of the solution for a work being written is to include a test script at the end of each section, and to place a master test script (like "test all") at the top of the source. But different designers will prefer different approaches, and this testing system is no more than an optional convenience.

Many tests will only be sensible in given places, which may be hard to reach from the initial position; or with the aid of given things, which may be difficult to obtain. We are therefore allowed to add stipulations to test scripts:

`Test balloon with "get balloon / blow balloon / drop balloon" holding the balloon.`

The "... holding the balloon" means that the balloon will be transferred to the player's ownership immediately before the test script is run, unless it is already held. Similarly:

`Test jam with "get jam / taste jam / eat jam" in the Kitchen.`



Or we might want to say both:

Test jam with "get jam / taste jam / eat jam" in the Kitchen holding the jam.




(Single quotation marks in test scripts are interpreted the same way in test scripts as they are in other text: that is, they are sometimes read as double-quotes unless they appear to be present as apostrophes. The notation ['] forces a single quotation mark if necessary. Similarly, [/] forces a literal forward slash, and prevents the / from being read as dividing up two commands.)

Sometimes when testing it's convenient to get hold of something not easily available at the moment. The testing command "PURLOIN" does this:

The jewelled Turkish clockwork hat is in the sealed glass box.

```
> PURLOIN HAT
[Purloined.]
```

This can also make test scripts shorter, but of course it's important to make sure that people without PURLOIN powers can still play through.

- 
-  Start of Chapter 2: The Source Text
  -  Back to §2.7. The SHOWME command
  -  Onward to §2.9. Material not for release
- 

## §2.9. Material not for release

Special testing commands, like "TEST" and "SHOWME", are automatically excluded from the story if it is exported from the Inform application using the Release button. We sometimes want to write our own for-testing-purposes-only code, though, and for this purpose we are allowed to designate whole headings as being "not for release":

Section 10 - Open sesame - Not for release

```
Universal opening is an action applying to nothing.
Understand "open sesame" as universal opening.
Carry out universal opening: now all doors are open.
Report universal opening: say "Open Sesame!"
```




Clearly we do not wish the final reader to be able to type "OPEN SESAME", so this whole heading will be disregarded in the Release version, as will any heading whose name includes "not for release".

Note that if a chapter, say, is marked as "not for release", then its subheadings (mere sections) will also not be for release. If in doubt, check the "Contents" index: if any section is "not for release" then so are all of its subheadings.

The reverse effect is produced by:

That is, it marks material included only in a Release version.

---

-  Start of Chapter 2: The Source Text
  -  Back to §2.8. The TEST command
  -  Onward to §2.10. Installing extensions
- 

## §2.10. Installing extensions

The original Inform of 1993 provided no special facilities for "extensions" - in effect, additional packets of rules providing extra features - but the creation and circulation of these extensions soon became a flourishing part of Inform culture. Today's Inform actively promotes sharing of such extensions, both to bring writers together and to support good practice. For the user of an extension, the advantage is clear: why go to great trouble to (say) work out how to make doors open automatically as needed, when somebody else has already perfected this? For the writer of an extension, there is the satisfaction of producing a good solution to a ticklish problem, and contributing to the public good.

Newcomers will probably not need extensions for quite some while, but there is nothing difficult about using them, so a few brief notes are worth giving here. (The final chapter of the documentation covers the writing of new extensions.)

Extensions are identified by name (say "Following People") and also by author (say "Mary Brown"). They need to be installed before they can be used, which means downloading them from the Internet. By far the easiest way to do this is to use the Public Library feature of Inform: then the application can do everything, letting us either choose individual extensions or download them en masse. But it's also possible to install extensions by hand.

When using Inform on Mac OS X, use the File menu item **Show Extensions Folder** to open the relevant folder in the Finder. Each author has a subfolder of this folder, and his or her extensions live inside it.

When using Inform on Windows, this means storing them in the folder

[My Documents\Inform\Extensions](#)

Each author has a subfolder of this folder, and his or her extensions live inside it. Our example extension should therefore be placed as:

[My Documents\Inform\Extensions\Mary Brown\Following People.i7x](#)

When using Inform on Linux, this means storing them in the folder

[~/Inform/Extensions/](#)

where "~" signifies your home folder. (This will have been created for you the first time you ran i7.) Each author has a subfolder of this folder, and his or her extensions live inside it. Our

example extension should therefore be placed as:

[~/Inform/Extensions/Mary Brown/Following People.i7x](#)

When using Inform on Linux, this means storing them in the folder

[~/Inform/Extensions/](#)

where "~" signifies your home folder. (This will have been created for you the first time you ran i7.) Each author has a subfolder of this folder, and his or her extensions live inside it. Our example extension should therefore be placed as:




[~/Inform/Extensions/Mary Brown/Following People.i7x](#)

In fact, though, Inform can automatically install extensions for us: we need only select the "Install Extension..." item on the File menu.

The actual extension file should always be named with a ".i7x" suffix, meaning "I7 extension" - for instance, "Following People.i7x".

To provide an example, Emily Short's useful extension "Locksmith" is one of a small number of extensions which come ready-installed as part of the basic Inform package, and need not be downloaded and installed.

Each time that Inform translates any source text, it performs a quick check of the extensions available, and updates its own internal records. A directory of the extensions currently installed can be found by clicking on "Installed Extensions" from the Extensions panel. This is also worth visiting in order to browse the Public Library, a selection of extensions contributed by Inform users.

- 
-  Start of Chapter 2: The Source Text
  -  Back to §2.9. Material not for release
  -  Onward to §2.11. Including extensions
- 

## §2.11. Including extensions

We talk about "including" such an extension into a work of IF because the process merges rules and behaviours from the extension with those we have described ourselves. It's not uncommon for contributions by five or six different people to be pooled together this way.




Including an extension is only a matter of writing a single sentence in the source. For instance:

[Include Locksmith by Emily Short.](#)

Note that it is compulsory to name both extension and author.

Many extensions come with their own documentation. Again, follow the "Installed Extensions" link to see what's available from them.

---

-  [Start of Chapter 2: The Source Text](#)
  -  [Back to §2.10. Installing extensions](#)
  -  [Onward to §2.12. Use options](#)
- 

## §2.12. Use options

One more preliminary. Inform has a small number of optional settings which affect the result of translating the source. The sentence:

[Use American dialect.](#)

makes the resulting work of IF use American spellings (except where the designer spells otherwise) and the American convention for spelling out numbers (thus, "one hundred seventeen" not "one hundred and seventeen"). Similarly:

[Use the serial comma.](#)

uses a comma when printing lists: thus "Julian, Dick, George, and Anne" rather than "Julian, Dick, George and Anne". A more profound change is made by

[Use scoring.](#)

which introduces the concept of a numerical score - something which modern authors of interactive fiction often feel is inappropriate, which is why Inform only provides it on request. Two alternative options:

[Use full-length room descriptions.](#)

[Use abbreviated room descriptions.](#)

change the normal way room descriptions are shown: normally they are given in full, but in abbreviated mode, they're never given. (The latter is a bad idea in any publicly released story, but is provided for completeness and in case it may help testing.) Alternatively, we can set the traditional Infocom-style of room description to any of VERBOSE, BRIEF and SUPERBRIEF:

[Use VERBOSE room descriptions.](#)

[Use BRIEF room descriptions.](#)

[Use SUPERBRIEF room descriptions.](#)

The default is now VERBOSE, but until 2010 it was BRIEF.

Next we have:




[Use undo prevention.](#)

which disables the UNDO verb, both in play and after death, for the benefit of stories which are heavily randomised and where we do not want players to keep on UNDOing until they get a random outcome which is to their taste. (Many players consider UNDO to be their birthright, and that any work using this option is an abomination: indeed, it has even been suggested that this section of the Inform documentation be censored. To use the option is to court controversy if not outright hostility.)

We can combine any number of options in a single "Use" sentence, so for example:

[Use American dialect and the serial comma.](#)

brings about both of these changes.

- 
-  [Start of Chapter 2: The Source Text](#)
  -  [Back to §2.11. Including extensions](#)
  -  [Onward to §2.13. Administering classroom use](#)
- 

## §2.13. Administering classroom use

Inform is increasingly used in education, where teachers sometimes need to install it on a whole room of computers at once, and want to monitor their students' progress. There is no special "classroom" version of Inform, but a couple of small administration features in the standard Inform - usually never needed - might be helpful to teachers.

When Inform starts up, it now looks for a file called Options.txt inside the user's home folder for Inform. (On Mac OS X, this is "~/Library/Inform"; on Windows, "My Documents\Inform", and so on.) If the file is present, then the text in it is added to the source text of everything Inform translates.

This must be used only to set use options, specify test commands, and give release instructions. For example, the following is a valid "Options.txt":

```
Use American dialect.  
Test fish with "fish/fish with pole/angle".  
Release along with source text.
```

The idea is that this file can be used for setting up a standard configuration on multiple machines in a classroom setting. Here the instructor can make sure the Release button will do what she would like, and can arrange for each student's copy of Inform to respond to given Test commands: for instance, if the class has an assignment to create a simulation of a camera, the instructor could set up "Options.txt" so that TEST CAMERA would run through some commands the camera ought to respond to.

A new use option, "Use telemetry recordings.", causes Inform to copy its outcome and problem messages to files in its home folder (see above) as they occur. These files are dated, so that for instance

[Telemetry 2009-03-25.txt](#)

contains all of the recorded activity on 25 March 2009. Telemetry only records the contents of the "Problems" panel - notes of success or failure, and problem messages - and nothing is transmitted via any network, so it isn't really surveillance. The user can deliberately add a note to the current telemetry file by writing something like this in source text:




\* "I don't get it! What's a kind? Why can't the lamp be lighted?"

(This is a way to make a note for the benefit of someone who will read the telemetry file - for instance, to comment on a problem message that has just appeared. Note the double-quotes. Otherwise, it's meant to look like the standard way that beta-testers mark up IF transcripts.)

These two features have been added in response to requests from education users. Let's suppose that Mr Lebling, who teaches 5th grade in Minnesota, wants to set things up just right for his class. He installs Inform on the ten computers they will use, and also copies an Options.txt file from his memory stick onto each one. The Options.txt file reads:

Use serial comma.  
Use American dialect.  
Use telemetry recordings.

Now Mr Lebling's class won't be confronted with English spellings, and so on. And most of the kids are happy, but Mr Lebling gets the feeling that young Marc wasn't really paying attention, so after class he checks that day's Telemetry file for that computer to see what Marc was up to, and whether he was stuck on something.

- 
-  Start of Chapter 2: The Source Text
  -  Back to §2.12. Use options
  -  Onward to §2.14. Limits and the Settings panel
- 

## §2.14. Limits and the Settings panel

No computer has unlimited capacity, and a large, complex project may eventually bump its head against the ceiling.

Inform is a system for translating textual descriptions of interactive fiction into "story files". No single format of story file is standard to the IF community. The formats developed over the history of IF differ in three key respects:

- the range of computers or devices capable of playing them;
- how large they are, that is, how much play they can express;
- what extra-textual effects they can bring off.

Inform can write to two different formats. Neither of these is proprietary, and neither was created by the authors of Inform: each format is a community property, defined by published standards documents. An individual Inform project can make its own choice of story file format, using that project's Settings panel.

Newly created projects are set up with the Glulx format. This has largely taken over from an earlier format called the Z-machine, but Inform can still generate a version 8 Z-machine file (a so-called "z8") if required. The Z-machine is of historic importance, and may continue to be useful for certain tasks where Glulx support is not yet available, but most users will want to keep the Glulx format set all of the time.

Internally, the Inform application uses a tool called Inform 6 (which was once the entire Inform system) to manufacture the story file. There are therefore two ways that large projects can run out of space:

- (a) By exceeding some maximum in Inform 6, or
- (b) By exceeding some fundamental limitation of the current story file format.

In both cases, the Inform application will display a Problems page explaining that the Inform 6 tool has failed to work as intended, and refer us to the "console output" - the text produced by Inform 6 - which is normally ignored, but can be found on the Console tab of the Results panel.

In case (a), Inform 6 will say that a memory setting has been exceeded: it will say what this setting is called (for instance "MAX\_ZCODE\_SIZE") and what its current value is (for instance 50000). We can then avoid the problem by adding the following use option into the source text:

Use MAX\_ZCODE\_SIZE of 60000.

And similarly for every other Inform 6 memory setting. (If the source tries to specify the same setting more than once - which is quite possible if extensions are included, with rival ideas - then the highest value is used.)

Case (b) is only likely to happen with the Z-machine format, since Glulx has a huge capacity; so the cure here is to switch to Glulx in the Settings. But if that's not possible for some reason - say, if we want a story file playable on a tiny handheld computer unable to manage Glulx - we still have a few options. Unless the story is very large (in which case there is little we can do), the "z8" format is most likely to be exhausted for lack of what is called "readable memory", with a message like so:

This program has overflowed the maximum readable-memory size of the Z-machine format. See the memory map below: the start of the area marked "above readable memory" must be brought down to \$10000 or less.




followed by a tabulation of how the Z-machine's storage has been used, a large but not very useful diagram. The first time one runs into the problem on a large project, it can be postponed, by adding the following to the source:

Use memory economy.

(Economy cuts down the verbosity of some of the testing commands, but otherwise subtracts no performance.) Writing this into the source is the equivalent of a diver switching to an emergency oxygen tank: it gives us a generous safety margin, but also tells us that now is the time to wrap things up.

If we hit the problem again, genuine cuts must be made. As a general rule, the most memory-expensive ingredients of an Inform design are various-to-various relations between large kinds such as "thing" or, if there are many rooms, "room". Other than that, if a kind has been festooned with new properties and we have created dozens of items of that kind, then we can get a fairly large saving simply by doing without one of those properties; and so on.

The ultimate memory-saving device, of course, is the one used by book publishers when there are too many pages to bind: to cut the design into two stories, Part I and Part II.

- 
-  Start of Chapter 2: The Source Text
  -  Back to §2.13. Administering classroom use
  -  Onward to §2.15. What to do about a bug
- 

## §2.15. What to do about a bug

In its present guise, Inform is a young piece of software, and bugs are to be expected from time to time. The most obvious bugs are the ones which Inform catches itself, when it confesses that it has halted in failure, or translated the source text into a program which cannot be compiled further. But sometimes it will also happen that Inform will issue a misleading Problem message, or appear to work normally but to produce a story which does not do what it should have done.

It is very helpful for users to report faults, so that the program can be improved for everyone else. To report a fault, please first check with the Inform home page to make sure that the version of Inform you have used to detect the fault is the latest version available. You can find the latest versions at

<http://inform7.com/download/>




If the bug is still present in the latest version, please report the bug using Inform's bug tracking database, found at

<http://inform7.com/mantis/>

We can search existing bug reports using the search box at

[http://inform7.com/mantis/view\\_all\\_bug\\_page.php](http://inform7.com/mantis/view_all_bug_page.php)

It may be that someone else has already identified the bug and even that a workaround for users is suggested. If not, please make an account at the bug tracking system and submit the requested information to help Inform's maintainers track and fix the fault.

- 
-  Start of Chapter 2: The Source Text
  -  Back to §2.14. Limits and the Settings panel
  -  Onward to §2.16. Does Inform really understand English?
-



## §2.16. Does Inform really understand English?

No. No computer does, and Inform does not even try to read the whole wide range of text: it is a practical tool for a particular purpose, and it deals only with certain forms of sentence useful to that purpose. Inform source text may look like "natural language", the language we find natural among ourselves, but in the end it is a computer programming language. Many things which seem reasonable to the human reader are not understood by Inform. For instance, Inform understands

something which is carried by the player

but not (at present, anyway)

something which the player carries

even though both are perfectly good English. So it is not always safe to assume that Inform will understand any reasonable instruction it is given: when in doubt, we must go back to the manual.

More philosophically, to "understand" involves contextual knowledge. Just because Inform recognises and acts on a sentence, does it really understand what we meant? It will turn out that Inform is both good and bad at this. For instance, from

Mr Darcy wears a top hat.

Inform will correctly deduce that Darcy is a person, because inanimate objects do not ordinarily wear clothes, and that the top hat is clothing. But it will not automatically know that Darcy is a man rather than a woman because it does not know the social convention implied by "Mr". Moreover, if instead we had written

Mr Darcy carries a top hat.

then Inform would not guess that the top hat is clothing. This is because it does not have the vast vocabulary and experience of a human reader: it is probably discovering the word "hat" for the first time.

Finally, it is best to avoid ambiguities rather than rely on Inform to know which meaning is patently absurd. For instance, in

Heatwave bone breaks clog hospital.




(a headline once printed by the *Oxford Mail* newspaper) a human reader quickly realises that there is no clog hospital being broken. But if Inform had been taught the verbs *to break* and *to clog* then that is exactly the conclusion it would have drawn. Or an example which genuinely arose in beta-testing:

The life support unit fits the egg.

in which Inform construed the verb as *support* and not *fits*, and then created items called "the life" (plural) and "unit fits the egg".





That disclaimer completes the groundwork, and we are ready to begin on simulating a world to explore.

---

-  [Start of Chapter 2: The Source Text](#)
  -  [Back to §2.15. What to do about a bug](#)
  -  [Onward to Chapter 3: Things: §3.1. Descriptions](#)
-

## Chapter 3: Things

§3.1. Descriptions; §3.2. Rooms and the map; §3.3. One-way connections;  
§3.4. Regions and the index map; §3.5. Kinds; §3.6. Either/or properties;  
§3.7. Properties depend on kind; §3.8. Scenery; §3.9. Backdrops;  
§3.10. Properties holding text; §3.11. Two descriptions of things; §3.12. Doors;  
§3.13. Locks and keys; §3.14. Devices and descriptions; §3.15. Light and darkness;  
§3.16. Vehicles and pushable things; §3.17. Men, women and animals;  
§3.18. Articles and proper names; §3.19. Carrying capacity;  
§3.20. Possessions and clothing; §3.21. The player's holdall; §3.22. Food;  
§3.23. Parts of things; §3.24. Concealment; §3.25. The location of something;  
§3.26. Directions

-  Contents of *Writing with Inform*
-  Chapter 2: The Source Text
-  Chapter 4: Kinds
-  Indexes of the examples

### §3.1. Descriptions

At its simplest, the interactive fiction will be simulating a physical world to explore. The forerunner of today's IF is generally agreed to be a computer simulation by Will Crowther of the exploration of a cave system in the Mammoth and Flint Ridge chain of caves in Kentucky, a part of which might be described in Inform thus:

"Cave Entrance"

The Cobble Crawl is a room. "You are crawling over cobbles in a low passage. There is a dim light at the east end of the passage."

A wicker cage is here. "There is a small wicker cage discarded nearby."

The Debris Room is west of the Crawl. "You are in a debris room filled with stuff washed in from the surface. A low wide passage with cobbles becomes plugged with mud and debris here, but an awkward canyon leads upward and west. A note on the wall says, 'Magic word XYZZY'."

The black rod is here. "A three foot black rod with a rusty star on one end lies nearby."

Above the Debris Room is the Sloping E/W Canyon. West of the Canyon is the Orange River Chamber.

Here we sketch in four of Crowther's locations, and two objects: just enough to be able to walk around the caves and pick up the rod and the cage. The text in quotation marks will appear verbatim as paragraphs shown to the player as the caves are explored. The first paragraph, as we have seen, is the title of the work. The other quotations describe the places and objects introduced.

If we play this story, we find that we can type TAKE CAGE or TAKE WICKER CAGE, for instance, but not TAKE SMALL CAGE. Inform saw that we called this "a wicker cage" when it first appeared in the source text, and assumed that the player would call it that, too. (Whereas it didn't look inside the descriptive text to allow for TAKE SMALL CAGE or TAKE DISCARDED CAGE or TAKE NEARBY CAGE.) A small limitation here is that probably only the first 9 letters of each word are read from the player's command. This is plenty for handling the wicker cage and the black rod, but it might be embarrassing at a meeting of the Justice League to find that KISS SUPERHERO and KISS SUPERHEROINE read as if they are the same command.

So we have already found that Inform has made some assumptions about what we want, and imposed some limitations on how much computational effort to go to when the work of IF is finally played. If Inform guesses what we need wrongly, we need to know more advanced features of the language in order to overcome these problems. (We shall see how to change the way the player's commands are read in the chapter on Understanding.)











This is often how Inform works: make the standard way of doing things as simple as possible to describe, but allow almost any behaviour to be altered by more elaborate source text. As an example of that, the player begins in the Cobble Crawl because it was the first room created in the source text, but we could instead have written text like:

The player is in the Cobble Crawl.

to override that. This can make the source text easier to follow if the rooms are sometimes being created in a less obvious way. For example, if we write:

The silver bars are in the Y2 Rock Room.  
The Cobble Crawl is a room. South of the Crawl is Y2.

then the first room to be created will actually be the Y2 Rock Room, so that's where the player will be starting unless we say otherwise.

- 
-  Start of Chapter 3: Things
  -  Back to Chapter 2: The Source Text: §2.16. Does Inform really understand English?
  -  Onward to §3.2. Rooms and the map
  -  Example 2:  **Bic** Testing to make sure that all objects have been given descriptions.
  -  Example 3:  **Verbosity 1** Making rooms give brief room descriptions when revisited.
  -  Example 4:   **Slightly Wrong** A room whose description changes slightly after our first visit there.
- 

## §3.2. Rooms and the map

Rooms are joined together at their edges by "map connections", most of which are pathways in one of the eight cardinal compass directions: north, northeast (written without a hyphen), east, southeast, south, southwest, west, northwest. We also have up and down, suitable for staircases or ladders. In real life, people are seldom conscious of their compass bearing when

walking around buildings, but it makes a concise and unconfusing way for the player to say where to go next, so is generally accepted as a convention of the genre.

Two more directions are provided by Inform: "inside" and "outside". These are best used when one location is, say, a meadow and the other is a woodcutter's hut in the middle of it; we might then say

Inside from the Meadow is the woodcutter's hut.

The "from" is important, as it clarifies that we intend to link two different locations, not to create an item - the hut - in a single location - the meadow.

A problem which sometimes arises when laying out maps is that Inform allows short forms of room names to be used as abbreviations. This is usually a good idea, but has unfortunate results if we write:

The Airport Road is west of the Fish Packing Plant. The Airport is west of the Airport Road.

...because "Airport" is taken as a reference to "Airport Road", so Inform makes only two locations, one of which supernaturally leads to itself. We can avoid this by writing:

The Airport Road is west of the Fish Packing Plant. A room called the Airport is west of the Airport Road.

Using "called" is often a good way to specify something whose name might give rise to confusion otherwise. It always makes something new, and it is also neatly concise, because we can establish something's kind and name in the same sentence. As another example, suppose we want to create a room called "South of the Hut", to south of the Hut. We can't do so like this:

South of the Hut is a room. South of the Hut is south of the Hut.

...because Inform will read that first sentence as placing a (nameless) room to the south of a room called "Hut". Once again "called" can save the day:

South of the Hut is a room called South of the Hut.

It is best to use "called" in the simplest way possible, and in particular, best not to use "called" twice in the same sentence. Consider:













The kitchen cabinet contains a container called a mixing bowl and a portable supporter called a platter.

It is unlikely that anyone would want to name something "a mixing bowl and a portable supporter called a platter", but not impossible, and Inform tends not to be a good judge of what is likely.

(If we really want to get rid of this issue once and for all, starting the source text with the use option "Use unabbreviated object names." will do it, but the effect is drastic. This instructs Inform not to recognise names other than in full. For example:

West of the Kitchen is the Roaring Range. South of the Range is the Pantry.

is ordinarily read by Inform as constructing three rooms (Kitchen, Roaring Range, Pantry); but with this use option set, it makes four (Kitchen, Roaring Range, Range, Pantry), in two disconnected pieces of map. Handle with care.)

- 
-  Start of Chapter 3: Things
  -  Back to §3.1. Descriptions
  -  Onward to §3.3. One-way connections
  -  Example 5:  **Port Royal 1** A partial implementation of Port Royal, Jamaica, set before the earthquake of 1692 demolished large portions of the city.
  -  Example 6:   **Up and Up** Adding a short message as the player approaches a room, before the room description itself appears.
  -  Example 7:    **Starry Void** Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.
- 

### §3.3. One-way connections

Connections are ordinarily two-way, but do not have to be. One of the map connections in the Mammoth Cave simulation was made by the sentence:

The Debris Room is west of the Crawl.

Besides reading this sentence at face value, Inform also deduced that the Crawl was probably meant to be east of the Debris Room: in other words, that the path between them is a two-way one. When Inform makes guesses like this, it treats them as being less certain than anything explicitly stated in the source. Inform will quietly overturn its assumption if information comes to hand which shows that it was wrong. That might happen in this case if another sentence read:

The Hidden Alcove is east of the Debris Room.

These two sentences are not contradictory: Inform allows them both, simply accepting that the world is more complicated than it first assumed. There are relatively few situations where Inform has to make educated guesses, but when it does, it tries always to follow Occam's Razor by constructing the simplest model world consistent with the information in the Source text.

We can even explicitly make a route which turns around as it leads between two rooms:

West of the Garden is south of the Meadow.








If we want to establish a route which cannot be retraced at all, we can specify that a particular direction leads nowhere:

East of the Debris Room is nowhere.

Finally, note that Inform's assumptions about two-way directions are only applied to simple sentences. When the source text seems to be saying something complicated, Inform takes it as a precise description of what's wanted. So, for example, in:

[The Attic is above the Parlour.](#)  
[The Attic is a dark room above the Parlour.](#)

Inform makes guesses about the first sentence, and makes a two-way connection; but it accepts the second sentence more precisely, with just a one-way connection.

- 
-  Start of Chapter 3: Things
  -  Back to §3.2. Rooms and the map
  -  Onward to §3.4. Regions and the index map
  -  Example 8:  **Port Royal 2** Another part of Port Royal, with less typical map connections.
  -  Example 9:  **The Unbuttoned Elevator Affair** A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.
- 

## §3.4. Regions and the index map

Rooms represent individual places to which one can go, but we tend to think of the world around us in larger pieces: we think of a house and a garden, rather than each of the single rooms of the house and all corners of its garden. To Inform a collection of rooms is called a "region", and we can create one like so:

[The Arboretum is east of the Botanical Gardens. Northwest of the Gardens is the Tropical Greenhouse.](#)

[The Public Area is a region. The Arboretum and Gardens are in the Public Area.](#)

The real usefulness of creating regions like "Public Area" will only appear later, when we begin defining rules of play which apply in some areas but not others, but in the mean time we can see the effect by turning to the World tab of the Index. In the World Index, Inform draws a map - or at least a stylised attempt at a diagram of the rooms and their connections: this will not always correspond to how we imagine things, but with any luck it should mostly be right.

Rooms are represented by coloured squares, and the colour-coding is done by region. In the above example, the two "Public Area" rooms are coloured green (as it happens); the Greenhouse, since it belongs to no region, is a neutral grey.

Regions can be put inside each other:

[The University Parks is a region. The Public Area is in the University Parks.](#)

but they are not allowed to overlap other than by one being entirely inside the other.

★ See **Improving the index map** for ways to adjust the way the index map is drawn or exported for publication

---

- ↑ Start of Chapter 3: Things
  - ← Back to §3.3. One-way connections
  - Onward to §3.5. Kinds
  - ↓ Example 10: ★ **Port Royal 3** Division of Port Royal into regions.
- 

## §3.5. Kinds

The following description runs to only 33 words, but makes a surprisingly intricate design. It not only places things within rooms, but also places them very specifically with respect to each other:

"Midsummer Day"

East of the Garden is the Gazebo. Above is the Treehouse. A billiards table is in the Gazebo. On it is a trophy cup. A starting pistol is in the cup.

Inform needs to identify the places and objects being described by the nouns here, and to guess what it can about them. For instance, the pistol can be picked up but not walked inside, whereas the Treehouse is the reverse. (This is obvious to someone who knows what these words mean, less obvious to a computer which does not, but the text contains sufficient clues.) Inform does this by sorting the various nouns into different categories, which are called "kinds". For instance:

Garden, Gazebo, Treehouse - **room**  
billiards table - **supporter**  
cup - **container**  
starting pistol - **thing**  
East, up (implied by "above") - **direction**

(A container is something which can contain other things, and a supporter similarly.) For instance Inform knows that if one thing is in another, then the second thing is either a room or a container, and if one thing is on another, the second thing is a supporter. This worked nicely for the design above, but:

In the Treehouse is a cardboard box.

results in the cardboard box being made only a "thing": because nothing has been put inside it, there is no reason for Inform - which does not know what a cardboard box looks like - to guess that it is a "container". So we need to add:








The box is a container.

It is rather clumsy to have to write two sentences like this, so we would normally write this instead:



In the Treehouse is a container called the cardboard box.

---

-  Start of Chapter 3: Things
  -  Back to §3.4. Regions and the index map
  -  Onward to §3.6. Either/or properties
  -  Example 11:  **First Name Basis** Allowing the player to use different synonyms to refer to something.
  -  Example 12:  **Midsummer Day** A few sentences laying out a garden together with some things which might be found in it.
- 

### §3.6. Either/or properties

Some containers, like bottles, can be opened: others, like buckets, cannot. If they can be opened, then sometimes they will be open, and sometimes closed. These are examples of properties, which can change during play. The following source sets some properties:

The cardboard box is a closed container. The glass bottle is a transparent open container. The box is fixed in place and openable.

There are only four different properties referred to here. Closed means not open, and vice versa, so these two adjectives both refer to the same property. (As might be expected, when a container is open, one can see inside and place things within, or take them out.) The glass bottle and the box being containers is a matter of their kinds, which is something fundamental and immutable, so "container" does not count as a property.






A "transparent" container is one which we can see inside even when it is closed, and the opposite is an "opaque" container.

The property of being "fixed in place" ensures that the player cannot pick the item up and walk away with it: this is useful for such things as oak trees or heavy furniture. The opposite condition is to be "portable".

A container which is "openable" can be opened or closed by the player; as might be expected, the opposite is "unopenable".

With a really large cardboard box, we might imagine that the player could get inside: such a container should be declared "enterable".

---

-  Start of Chapter 3: Things
  -  Back to §3.5. Kinds
  -  Onward to §3.7. Properties depend on kind
  -  Example 13:  **Tamed** Examples of a container and a supporter that can be entered, as well as nested rooms.
-

## §3.7. Properties depend on kind






Properties depend very much on kind. It makes no sense to ask whether a room is transparent or opaque, for instance, so Inform will not allow this either to be specified or queried.

Another way that kind influences properties can be seen from an earlier example:

The Gazebo is a room. A billiards table is in the Gazebo. On it is a trophy cup. A starting pistol is in the cup.

The cup, the pistol and the table are all allowed to have the "fixed in place" property, but in fact only the table actually has it: the cup and the pistol are created as "portable" instead. This is because Inform knows that most things are portable, but that supporters - such as the table - are usually fixed in place. If this assumption is wrong, we need only add the line:

The table is portable.

- 
-  Start of Chapter 3: Things
  -  Back to §3.6. Either/or properties
  -  Onward to §3.8. Scenery
  -  Example 14:  **Disenchantment Bay 1** A running example in this chapter, Disenchantment Bay, involves chartering a boat. This is the first step: creating the cabin.
- 

## §3.8. Scenery

As we have just seen, making something "fixed in place" will prevent it from being picked up or moved. But it remains substantial enough to be described in its own paragraph of text when the player visits its location. This can be unfortunate if it has also been described already in the body of the main description for that location. For instance, if we wrote:

The Orchard is a room. "Within this quadrille of pear trees, a single gnarled old oak remains as a memory of centuries past." The gnarled old oak tree is fixed in place in the Orchard.

This would end up describing the oak twice, once in the paragraph about the Orchard, then again in a list of things within it:

### **Orchard**

Within this quadrille of pear trees, a single gnarled old oak remains as a memory of centuries past.

You can see a gnarled old oak tree here.

We avoid this by making it "scenery" instead of "fixed in place":

The gnarled old oak tree is scenery in the Orchard.

Any thing can be scenery, and this does not bar it from playing a part in the story: it simply means that it will be immobile and that it will not be described independently of its room. Being immobile, scenery should not be used for portable objects that are meant to be left out of the room description.

If a supporter is scenery, it may still be mentioned in the room description after all, but only as part of a paragraph about other items, such as

On the teak table are a candlestick and a copy of the Financial Times.

If the player takes the candlestick and the Times, the teak table will disappear from mention. (Scenery containers do not behave in this way: their contents are assumed to be less immediately visible, and will be mentioned only if the player looks inside them.)

---

↑ Start of Chapter 3: Things

← Back to §3.7. Properties depend on kind

→ Onward to §3.9. Backdrops

↓ Example 15: ★ **Disenchantment Bay 2** Disenchantment Bay: creating some of the objects in the cabin's description.

↓ Example 16: ★ **Replanting** Changing the response when the player tries to take something that is scenery.

---

## §3.9. Backdrops

It is a cardinal rule that nothing can be in more than one place at the same time, but rules were made to be broken, and an exception is allowed for a special kind of thing called a "backdrop". For instance:

"Streaming"

The Upper Cave is above the Rock Pool.

The stream is a backdrop. It is in the Upper Cave and the Rock Pool.

Backdrops are ordinarily in the background: if the sky needed to be referred to in the course of play, it might be represented by a backdrop, for instance. Here we have a stream of water running through two rooms, though it might be any number. Backdrops are always fixed in place.

Backdrops can be put in regions as well as rooms, and if so, then they are present at every room in the given region (or regions), as well as any specific rooms they may also be put into. For instance:

The Outdoors Area is a region. The Moon is a backdrop. The Moon is in the Outdoors Area. The Moon is in the Skylight Room.

The special place "everywhere" can be given as the location of a backdrop to make it omnipresent:

The sky is a backdrop. The sky is everywhere.

Inform assumes that backdrops are also scenery unless told otherwise, so this will not result in messages like "You can also see the sky here." being included in room descriptions. In the case of the stream above, we could artfully mention it in passing in the room descriptions of the Upper Cave and the Rock Pool.

---

★ See **Moving backdrops** for ways to place backdrops in dynamically changing selections of rooms

---

↑ Start of Chapter 3: Things

← Back to §3.8. Scenery

→ Onward to §3.10. Properties holding text

↓ Example 17: **Disenchantment Bay 3** Disenchantment Bay: adding a view of the glacier.

---

## §3.10. Properties holding text

The properties we have seen so far have all been either/or: either open or closed, either transparent or opaque, either fixed in place or portable, either openable or not openable. However, some properties can have a much wider range of possibilities. For instance, the "description" of a room is the text revealed when the player first enters it, or types "look". This needs to be textual: Inform would complain if, for instance, we tried to set the description of something to the number 42. We have already seen a concise way to set the description of a room:

The Painted Room is north of the Undertomb. "This is the Painted Room, where strange wall drawings leap out of the dark at the gleam of your candle: men with long wings and great eyes, serene and morose."




This does the same thing as:

The Painted Room is north of the Undertomb. The description of the Painted Room is "This is the Painted Room, where strange wall drawings leap out of the dark at the gleam of your candle: men with long wings and great eyes, serene and morose."

Or even:

The Painted Room is north of the Undertomb. The description is "This is the Painted Room, where strange wall drawings leap out of the dark at the gleam of your candle: men with long wings and great eyes, serene and morose."

---

-  Start of Chapter 3: Things
  -  Back to §3.9. Backdrops
  -  Onward to §3.11. Two descriptions of things
- 

## §3.11. Two descriptions of things

The player's first sight of something is the text used as its "initial appearance":

The plain ring is here. "Cast aside, as if worthless, is a plain brass ring."

This text appears as a separate paragraph in the text describing the Painted Room. It will continue to be used until the first time player picks the ring up (if this ever happens), so it normally describes things in their original, undisturbed context. (Inform uses an either/or property called "handled" for this: something is "handled" if it has at some point been held by the player.)

Thus when a piece of text stands alone as a sentence in its own right, then this is either the "description" of the most recently discussed room, or the "initial appearance" of the most recently discussed thing. Either way, it is used verbatim as a paragraph in the text shown to the player visiting the room in question.








But a thing also has an ordinary "description", which is used to give a close-up look at it. This text is ordinarily only revealed to the player when a command like "examine ring" is keyed in:

The description of the plain ring is "No better than the loops of metal the old women use for fastening curtains."

---

★ See **Creating a scene** for the description of a scene, which is set in the same way

---

-  Start of Chapter 3: Things
  -  Back to §3.10. Properties holding text
  -  Onward to §3.12. Doors
  -  Example 18:  **Disenchantment Bay 4** Disenchantment Bay: fleshing out the descriptions of things on the boat.
  -  Example 19:  **Laura** Some general advice about creating objects with unusual or awkward names, and a discussion of the use of printed names.
- 

## §3.12. Doors

The map of an interactive fiction is the layout of rooms and the entrances and exits which connect them. So far, these map connections have always run from one room to another, like so:

The Painted Room is north of the Undertomb.

However, we can also interpose doors between rooms, like so:

The heavy iron grating is east of the Orchard and west of the Undertomb. The grating is a door.

The second sentence is needed since otherwise Inform will take "heavy iron grating" to be the name of a third room, whereas what we want is for the grating to be something physically present in both the Orchard and in the Undertomb, and acting as a conduit between them. To this end it needs to be a "door", a kind we have not so far seen. In the absence of any other instruction, a newly created door will be fixed in place, closed and openable.

The grating really does come in between the two rooms: the grating is what lies immediately east of the Orchard, not the Undertomb room. So if we wrote the following:

The Undertomb is east of the Orchard. The heavy iron grating is east of the Orchard and west of the Undertomb. The grating is a door.

then Inform would say that this is a contradiction: we said the Undertomb was east of the Orchard, but then we said that the grating was east of the Orchard.

Inform's "door" kind can be used for all manner of conduits, so the word door need not be taken literally. In Ursula K. Le Guin's beguiling novel "The Tombs of Atuan", from which the above rooms are stolen, it is not a grating which interposes, but:

The red rock stair is east of the Orchard and above the Undertomb. The stair is an open door. The stair is not openable.

In real life, most doors are two-sided, and can be used from either of the rooms which they join, but this is not always convenient for interactive fiction. Here is a one-sided door:

The blue door is a door. It is south of Notting Hill. Through it is the Flat Landing.

(Note the use of "it" here as an optional abbreviation.) This will make a door visible only on the Notting Hill side; no map connection will be made in the reverse direction, unless we ask for one.

So much for creating and describing individual doors. Once we need to write about doors in general, we are likely to want a way to find out where a given door sits in the map. The following phrases reveal this:

**front side of (object) ... room**

This phrase produces the first of the one or two rooms containing a door - first in the order given in the source text. Example: if

The red rock stair is east of the Orchard and above the Undertomb.

then "front side of the red rock stair" produces the Orchard. For a one-sided door, this produces the only room containing the door.

**back side of (object) ... room**

This phrase produces the last of the one or two rooms containing a door - last in the order given in the source text. Example: if

The red rock stair is east of the Orchard and above the Undertomb.

then "back side of the red rock stair" produces the Undertomb. A one-sided door has no "back side."

More often, we are dealing with a door and want to know what it leads to, but that depends where we're standing:

**other side of (door) from (room) ... object**

This phrase produces the room on the other side of the door, as seen from the given vantage point, which needs to be one of its sides. Example: if

The red rock stair is east of the Orchard and above the Undertomb.

then "other side of the red rock stair from the Undertomb" produces the Orchard, and vice versa.

**direction of (door) from (room) ... object**

This phrase produces the direction in which the door leads, as seen from the given vantage point, which needs to be one of its sides. Example: if













The red rock stair is east of the Orchard and above the Undertomb.

then "direction of the red rock stair from the Undertomb" produces up.

---

★ See **Adjacent rooms and routes through the map** for more phrases which can look at the current map layout

---

-  Start of Chapter 3: Things
  -  Back to §3.11. Two descriptions of things
  -  Onward to §3.13. Locks and keys
  -  Example 20:  **Disenchantment Bay 5** Disenchantment Bay: adding the door and the deck to our charter boat.
  -  Example 21:   **Escape** Window that can be climbed through or looked through.
  -  Example 22:    **Garibaldi 1** Providing a security readout device by which the player can check on the status of all doors in the game.
- 

### §3.13. Locks and keys

It seems unwise for a door in Notting Hill to be unlocked, so:

The blue door is lockable and locked. The matching key of the blue door is the brass Yale key.

Since the second sentence here is a little clumsy, we can equivalently say

The brass Yale key unlocks the blue door.









Yet a third way to say this is:

The blue door has matching key the brass Yale key.

This introduces three new properties: a door can be locked or unlocked; lockable or not lockable; and it can have a matching key, which must be another thing. The same thing can be the matching key of many different locks: and note that a door can be locked and even lockable without having a matching key at all, in which case the player trying to open it will be permanently out of luck. Doors are ordinarily unlocked, not lockable, and without a matching key.

Containers can also have locks, in exactly the same way, and are allowed to have the same properties. On the other hand supporters never have locks: it makes no sense to be able to lock a tabletop, for instance, and Inform will not allow any discussion of the matching key of a supporter, or of a supporter being locked or unlocked.

---

-  Start of Chapter 3: Things
  -  Back to §3.12. Doors
  -  Onward to §3.14. Devices and descriptions
  -  Example 23:  **Disenchantment Bay 6** Disenchantment Bay: locking up the charter boat's fishing rods.
  -  Example 24:   **Neighborhood Watch** A locked door that can be locked or unlocked without a key from one side, but not from the other.
-



## §3.14. Devices and descriptions

A "device" is another of the standard kinds of thing, and should be used for anything which can be switched on or off: a light switch, say, or a slide projector. Devices are generally machines, clockwork or electrical. A device is always either "switched on" or "switched off", but is switched off unless we specify otherwise.









That makes three kinds of thing which will likely change their appearance according to which of their two possible states they are in: doors and containers, which can be open or closed; and devices, which can be switched on or switched off. We would like to produce text accordingly, and we can do this using Inform's ability to make (almost) any piece of text change with circumstances. For instance:

The coffin is an openable container in the Undertomb. "[if open]The lid of a plank coffin yawns open.[otherwise]A plank coffin lies upon the dirt floor of the Tomb."

We could use a similar trick to make the appearance of a device change "if switched on". There will be much more about text substitutions, as instructions in square brackets like these are called, in later chapters.

★ See **Text with substitutions** for more on varying what is printed

---

-  Start of Chapter 3: Things
  -  Back to §3.13. Locks and keys
  -  Onward to §3.15. Light and darkness
  -  Example 25:  **Disenchantment Bay 7** Disenchantment Bay: making the radar and instruments switch on and off.
  -  Example 26:   **Down Below** A light switch which makes the room it is in dark or light.
- 

## §3.15. Light and darkness

Rooms can be "dark" or "lighted", though they are lighted by default, and are lighted in all the examples we have seen so far.

The Sinister Cave is a dark room. "A profoundly disquieting rock formation, apparently sculptured by some demonic hand, this is not a cave in which to relax."

When the player is in a dark room, he can still go in various directions, but he cannot see the room description or interact with any of the objects in the room, except those he is holding. This means that, unless we should change the Cave in some way during play, the text above ("A profoundly...") will only be read if the player succeeds in bringing light into the Cave, perhaps by bringing along the following:

The flaming torch is in the Sandy Passage. "Stuck loosely into the sand is a flaming torch." The flaming torch is lit.

A thing with the property of being "lit" will enable the player to see inside dark rooms, and to carry out other activities requiring light, such as examining items. A lit thing in an open container will still light up a room; a lit thing in a closed container will not, unless the container has been given the "transparent" property.

It is possible to adjust the way darkness behaves, and we will see more on this topic in the chapter on Activities.

★ See [Printing a refusal to act in the dark](#) for the first of several ways to control what is printed in the dark



Start of Chapter 3: Things



Back to §3.14. Devices and descriptions



Onward to §3.16. Vehicles and pushable things

---

## §3.16. Vehicles and pushable things

Next in the tour of standard kinds is the "vehicle". This behaves like (indeed, is) an enterable container, except that it will not be portable unless this is specified.

[In the Garage is a vehicle called the red sports car.](#)

The player can enter the sports car and then move around riding inside it, by typing directions exactly as if on foot: and the story will print names of rooms with "(in the red sports car)" appended, lest this be forgotten.

We have already seen that some things are portable, others fixed in place. In fact we can also make a third sort of thing: those which, although not portable, can be pushed from one room to another with commands like "push the wheelbarrow north". At a pinch, we might just be willing to allow:

[The red sports car is pushable between rooms.](#)

But of course this is a property which almost any thing can have, not just a vehicle. (Only "almost" because Inform will not allow a door to be pushable between rooms, in the interests of realism rather than surrealism.)




If we need vehicles which the passenger sits on top of, like a horse or a tractor, the standard "vehicle" kind will not be ideal. However, by loading one of the extensions which comes ready-installed:

[Include Rideable Vehicles by Graham Nelson.](#)

...we are provided with two more kinds, "rideable vehicle" and "rideable animal", just right for the tractor and the horse respectively. (As with all extensions, the documentation can be seen by clicking Go on some source which contains the above line, and then turning to the Contents index; or from the Installed Extensions tab of the Extensions panel.)

★ See *Going by, going through, going with* for further ways to customize vehicle behaviour

---

- ⬆ Start of Chapter 3: Things
  - ⬅ Back to §3.15. Light and darkness
  - ➡ Onward to §3.17. Men, women and animals
  - ⬇ Example 27:  **Peugeot** A journey from one room to another that requires the player to be on a vehicle.
  - ⬇ Example 28:  **Disenchantment Bay 8** Disenchantment Bay: a pushable chest of ice for the boat.
  - ⬇ Example 29:  **Hover** Letting the player see a modified room description when he's viewing the place from inside a vehicle.
- 

### §3.17. Men, women and animals

Rounding out the standard kinds provided by Inform are four for living things: "person", which is a kind of thing, and "man", "woman" and "animal", all kinds of person. For instance:

[In the Ballroom is a man called Mr Darcy.](#)

For the time being, men and women will be little more than waxworks: they will come to life only when we go beyond the present stage of creating an initial state of the world.






People can be male or female: this is an either/or property for the "person" kind, and it affects play at run-time a little, because the player can use "him" and "her" to refer to male or female people encountered. Men and women are always male and female respectively, and for animals we can choose either way, for example making a stallion male or a nanny goat female. Animals are male unless we say otherwise.

If our animal is instead something like a beetle or an earthworm, where gender doesn't seem to matter or even to exist, we can use the further property "neuter":

[The spider is a neuter animal in the Bathroom.](#)

The Standard Rules don't make people behave differently according to their genders, and the main difference comes down to language: whether we want the animal to be called "her", or "it". Because of the existence of "neuter", we sometimes need to be cautious about the use of the adjective "male": since Inform, partly for historical reasons, uses an either/or property for masculinity, neuter animals are also "male".

---

-  Start of Chapter 3: Things
  -  Back to §3.16. Vehicles and pushable things
  -  Onward to §3.18. Articles and proper names
  -  Example 30:  **Disenchantment Bay 9** Disenchantment Bay: enter the charter boat's Captain.
- 

## §3.18. Articles and proper names

Suppose we have said that:

In the Ballroom is a man called Mr Darcy.

When the Ballroom is visited, the man is listed in the description of the room as "Mr Darcy", not as "a Mr Darcy". This happened not because Inform recognised that Darcy is a proper name, or even because men tend to have proper names, but because Inform noticed that we did not use "a", "an", "the" or "some" in the sentence which created him. The following shows most of the options:

The Belfry is a room. A bat is in the Belfry. The bell is in the Belfry. Some woodworm are in the Belfry. A man called William Snelson is in the Belfry. A woman called the sexton's wife is in the Belfry. A man called a bellringer is in the Belfry.

In the Belfry is a man called the vicar. The indefinite article of the vicar is "your local".

In the resulting story, we read:

You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.

The subtlest rule here is in the handling of "the". We wrote "The bell is in the Belfry", but this did not result in the bell always being called "the" bell: in fact, writing "A bell is in the Belfry" would have had the same effect. On the other hand, "A woman called the sexton's wife is in the Belfry." led to the wife always being known as "the" sexton's wife, not "a" sexton's wife, because Inform thinks the choice of article after "called" shows more of our intention than it would elsewhere. These rules will never be perfect in all situations, so we are also allowed to specify indefinite articles by hand, as the vicar's case shows.

"Some" is worth a closer look, because English uses it in several different ways. By introducing the woodworm with "some", above, we established that it was plural. We might imagine that there are many worms, even though they are represented by a single thing in Inform. We can expect to see text in the story such as:

You can see some woodworm here.  
The woodworm are fixed in place.

But suppose we wanted something which there is an amount of, but which is not made up of individual items - a so-called mass noun like "water", or "bread". Now we can write:









The water is here. The indefinite article is "some".

and this time Inform does not treat the "some water" thing as a plural, so we might read:

You can see some water here.  
The water is hardly portable.

rather than "The water are hardly portable."

Finally, we can override these settings, if they still come out not as we intend, by explicitly changing the either/or properties "singular-named" (vs "plural-named") and "proper-named" (vs "improper-named").

- 
-  Start of Chapter 3: Things
  -  Back to §3.17. Men, women and animals
  -  Onward to §3.19. Carrying capacity
  -  Example 31:  **Belfry** You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.
  -  Example 32:   **Gopher-wood** Changing the name of a character in the middle of play, removing the article.
- 

## §3.19. Carrying capacity

The containers and supporters created so far have been boundlessly capacious: or rather, though we seldom notice the difference, have had a maximum carrying capacity of 100 items. This is clearly unrealistic for a small purse or a modest mantelpiece. We can impose upper limits with sentences like so:

The carrying capacity of the jewelled purse is 2.

The bijou mantelpiece has carrying capacity 3.

Attempts by the player to overfill, or overload, will now be rebuffed with a message such as "There is no room on the mantelpiece".

The player is not a container or a supporter, but nevertheless does have a carrying capacity: this is interpreted to mean the maximum number of items which can be carried at once.

The carrying capacity of the player is 4.

These restrictions only apply to the player (and other in-world characters): as the omnipotent creators, we are not restrained by them. Nothing prevents this:

The carrying capacity of the jewelled purse is 2. The diamond, the ruby and the sapphire are in the purse.

The player will be able to remove all three items, but only put two of them back. (This is probably something we only want very occasionally: perhaps to create a sack stuffed almost to bursting point.)

---



Start of Chapter 3: Things



Back to §3.18. Articles and proper names



Onward to §3.20. Possessions and clothing

---

## §3.20. Possessions and clothing

We have seen how to place objects in rooms, and in containers or on supporters. But what about people? Perhaps it could be said that they "contain" the fillings in their teeth, or "support" a top hat, but this is not very natural. Inform therefore never speaks of things being "in" or "on" people. Instead, they have two sorts of possessions: the things they carry, and the things they wear. (Body parts, such as arms and legs, are different again: see "parts" below for a clue to how to do these.) Thus:

*Mr Darcy wears a top hat. Mr Darcy carries a silver sword.*

In fact, Inform deduces from this not only who owns the hat and the sword, but also that Darcy has the kind "person", because only people can wear or carry.

As all the assertion verbs do, "to wear" and "to carry" have participles which Inform knows about. So we could equally well write:

*The scarlet coat is worn by Mr Wickham. The duelling pistol is carried by Mr Wickham.*

If we do not specify who does the wearing, or carrying, then this is assumed to be the player. Thus:

*A brass lantern and a rusty iron key are carried. The mosquito-repellent hat is worn.*






It would make no sense to "wear" the key, for instance, so Inform needs to distinguish between what is clothing and what is not. It does this with an either/or property called "wearable": if something has this property then the player will be allowed to wear it, provided it can first be picked up. Anything which is worn by somebody at the start of play is assumed to be wearable (unless we say otherwise). But if nobody is initially wearing the item in question, then we have to be explicit:

*The player carries a scarlet gown. The gown is wearable.*

(When we come to asking questions about the current situation, we will need to remember that "to carry" and "to wear" are different. Thus "if Lancelot carries the plate armour" will not be true if he is wearing it rather than carrying it under his arm. As we will later see, we can instead vaguely say "if Lancelot has the plate armour" to mean either carrying or wearing.)

★ *See **To carry, to wear, to have** for a more detailed explanation of carrying, wearing, and possessing as Inform understands them*

---

-  Start of Chapter 3: Things
  -  Back to §3.19. Carrying capacity
  -  Onward to §3.21. The player's holdall
  -  Example 33:  **Disenchantment Bay 10** Disenchantment Bay: things for the player and the characters to wear and carry.
- 

## §3.21. The player's holdall

When the player has only limited carrying capacity, play is likely to be tiresome, but we can make life easier by providing a way for the player to carry endless items without dozens of free hands to hold them all:

"Sackcloth"

The Attic is a room. The old blue rucksack is a player's holdall. The player is wearing the rucksack.

The carrying capacity of the player is 3.

In the Attic are a CD entitled *No Smoke Without Fire*, a 70s photograph of an American winning Wimbledon, a fraxinus branch, an urn holding your late great-aunt's remains, a convention badge from the American Society of Hypertension and a ghost story by M R James.






This example story introduces a new kind of container, the "player's holdall". This is a kind of which most stories will contain at most one example, but in principle there can be any number. A player's holdall is a capacious bag into which the player automatically places surplus items whenever his or her hands are full: trying the above example story and getting the items one by one will give the general idea.

Of course, if the carrying capacity of the player is never reached then there will never be any surplus items and a player's holdall will behave just like any other (portable, usually openable) container.

---

★ See **Units** for the tools to implement a more sophisticated capacity system

---

-  Start of Chapter 3: Things
  -  Back to §3.20. Possessions and clothing
  -  Onward to §3.22. Food
  -  Example 34:  **Disenchantment Bay 11** Disenchantment Bay: making a holdall of the backpack.
- 

## §3.22. Food

We have nearly reached the end of the chapter on Things, but one either/or property for things remains: every thing is either "edible" or "inedible". Unless we say otherwise, things are inedible. But for instance we might write:

The player carries a Macintosh apple. The Macintosh is edible.

(The type of computer is named after a variety of apple descended from a tree cultivated in 1811 by John McIntosh of Ontario.) Edible things are just like inedible ones, except that the player can EAT them. This will usually only consume the foodstuff in question, effectively destroying it, but using techniques from later chapters we could make the consequences more interesting.



Start of Chapter 3: Things



Back to §3.21. The player's holdall



Onward to §3.23. Parts of things

---

### §3.23. Parts of things

Everything has one and only one kind. This is both good and bad: good for clarity, bad if something needs to behave in two different ways at once. How might we simulate a car with an ignition key, given that no single thing can be both a "vehicle" and a "device" at the same time?

The Inform world model takes the view that such a car is too complicated to be simulated with a single thing. Instead it should be simulated as a vehicle (the car) which has a device (the ignition) attached. This is done using a third kind of containment to those seen so far ("in..." and "on..."): "part of".

#### "Buttons"

The Confectionary Workshop is a room. The Chocolate Machine is here. "The Chocolate Machine has pride of place. A lever and two buttons, one white, the other brown, seem to be the only controls. On top is a hopper."

A container called the hopper is part of the Chocolate Machine. The lever, the white button and the brown button are parts of the Chocolate Machine.

The Chocolatier's desk is here. "The Chocolatier evidently works at the imposing green-leather topped desk facing the Machine. It has three drawers with brass handles."

The upper drawer, the middle drawer and the lower drawer are parts of the desk. The upper drawer, the middle drawer and the lower drawer are openable closed containers. In the middle drawer is a sugared almond. In the lower drawer is a Battenburg cake. On the desk is a liquorice twist.










The cake, the twist and the almond are edible.



The machine and the desk each have several "parts" representing subsidiary pieces of themselves. The desk is a "supporter" (it needs to be, for the liquorice twist to be on top) but also has three "containers" attached, each of which can be opened or closed independently.

In the interests of realism, the standard rules of play protect these composite things. Thus if the desk were to be moved elsewhere (rolling on sugar casters perhaps) then its parts would move with it, and the player is not allowed to detach parts of things: the drawers can be opened or closed, but not pulled out altogether.

Note that rooms and regions are not allowed to have parts. (Rooms are already parts of regions, and to divide up rooms, we can either make several rooms or place containers or other obstacles in a single one.)

- 
-  Start of Chapter 3: Things
  -  Back to §3.22. Food
  -  Onward to §3.24. Concealment
  -  Example 35:  **Fallout Enclosure** Adding an enclosure kind that includes both containers and supporters in order to simplify text that would apply to both.
  -  Example 36:  **Brown** A red sticky label which can be attached to anything in the game, or removed again.
  -  Example 37:  **Disenchantment Bay 12** A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained.
- 

## §3.24. Concealment

Though realism can become tiresome in interactive fiction, there are times when we cannot go along with Inform's normal assumption that all of a person's possessions are visible to everybody else. People are not like containers, which either show all of their holdings or not, according to whether they are open or transparent. If a man is carrying a fishing rod and a wallet, one will be on open show, the other not. Some clothing is outwardly visible, but not all.

Whether or not something is concealed is not like the either/or properties we have seen so far - such as being "open" or "closed" - because it is not really a property of the thing itself, but depends on the habitual behaviour of its current owner. To talk about behaviour we have to use sentences of a kind not seen so far, and which will not fully be explained for some chapters to come.

But straightforward cases are easy to write, if only by imitating the following examples.

Here we make the Cloaked Villain invariably conceal anything she is holding or wearing:

[Rule for deciding the concealed possessions of the Cloaked Villain: yes.](#)

At which point we think about it more carefully, and then rewrite:

Rule for deciding the concealed possessions of the Cloaked Villain: if the particular possession is the sable cloak, no; otherwise yes.






(A rule which says neither "yes" nor "no" will decide yes, but it's best to spell out exactly what's wanted.)

Parts are treated exactly as if clothes or items being held, and the following will make the face and inscription on a coin invisible unless the player is holding it - the idea being that they are too small to be seen from farther away.

The coin is in the Roman Villa. The face and inscription are parts of the coin. Rule for deciding the concealed possessions of the coin: if the coin is carried, no; otherwise yes.

There is also an either/or property called "described"/"undescribed", intended to be used only as a last resort, but which has the ability to hide something from room descriptions. This not really hiding: the idea is that "undescribed" should be used only for cases where some other text already reveals the item, or where its presence is implicit. Even then, it should only be used when the item is intended to be taken or moved by the player at some point - if the item isn't intended to move, it's much better to make it "scenery". (There's only one commonly-found example - the player's own body, the "yourself", is undescribed.)

Note that the "undescribed" property is automatically removed from anything carried by, worn by or part of the player, even indirectly; and that nothing on top of an "undescribed" supporter will be visible in a room description, even if it itself is "described". (Scenery supporters don't suffer from that restriction, which is one reason scenery is a better option when possible.)

- 
-  Start of Chapter 3: Things
  -  Back to §3.23. Parts of things
  -  Onward to §3.25. The location of something
  -  Example 38:  **Search and Seizure** A smuggler who has items, some of which are hidden.
- 

## §3.25. The location of something

The model world created by Inform is partitioned into rooms. This means that everything which exists in the model world, exists in one of the rooms. If we write a sentence such as

[Professor Wilderspin is a man.](#)

and say nothing more about Wilderspin, then he does not physically exist at the start of the story: he is said to be "out of play", and stays that way until we move him into one of the rooms. A better metaphor might be that he is waiting in the wings, ready to come onto the stage.

Every thing is either out of play, or can be found in one of the rooms, and the property "location of X" gives us the room in question. The following condition tests, in effect, whether Wilderspin is in play:

if the location of Wilderspin is a room, ...

Which uses a new phrase:

**location of (object) ... room**

This phrase produces the room which, perhaps indirectly, contains the object given. Example: if the player stands in Biblioll College and wears a waistcoat, inside which is a fob watch, then

location of the fob watch

is Biblioll College. In general, a thing cannot be in two rooms at once, but there are two exceptions: two-sided doors, present on both sides, and backdrops. The "location of" a door is its front side, but a backdrop has no location. (Objects which are not things at all, such as rooms and directions, also have no location.)

We very often want to know the location of the player, and this is more simply called just "the location". (This is actually a value that varies rather than a phrase, but that's a technicality we can ignore here.)

The idea of indirect containment is useful enough to have a name: Inform calls it "enclosure". A thing encloses whatever is a part of itself, or inside itself, or on top of itself, and it also encloses anything that they enclose. And when something moves around, anything it encloses will move with it. In the example above, Biblioll College (a room) and the player (a person) both enclose the fob watch and the waistcoat. (The small print: a door is enclosed by the rooms on both sides; a backdrop is never enclosed.)

Enclosure is only useful when being used as a question. So the following is fine:

if the player encloses the fob watch, ...

But these will produce problem messages:

The player encloses the fob watch. The location of the trilobite is the Museum.

because they are too vague. Inform needs to know exactly where the fob watch and the trilobite will begin the story, whereas these sentences leave room for doubt about who or what is actually holding them.



Start of Chapter 3: Things




Back to §3.24. Concealment



Onward to §3.26. Directions



Example 39:  **Van Helsing** A character who approaches the player, then follows him from room to room.

---

## §3.26. Directions

"Direction" is a kind which is quite unlike most of those seen so far. While it has to do with the physical world, a direction does not exactly belong to it. One cannot find "southeast" sitting on a shelf. "Direction" is not a kind of thing, nor a kind of room: it is a kind in its own right.

Every direction has an "opposite" property, which is always another direction. These occur in matched pairs. The opposite of north is south, just as the opposite of south is north. The opposite of southeast is northwest, the opposite of inside is outside, and so on. When Inform reads a sentence like...

Bangkok is south of Nakhon Sawan.

...it assumes that the opposite map connection is probably also valid, so that

Nakhon Sawan is north of Bangkok.

The chapter began with the twelve directions built into Inform:

north, northeast, east, southeast, south, southwest, west, northwest, up, down, inside, outside

But the built-in set is not always appropriate. Sometimes this is too many; if we wanted to write about a Flatland, for instance, then up and down ought to go. But in practice it is better not to abolish them as directions but instead to forbid travelling in them. (See the Recipe Book for examples.)

But away from our familiar Earth, the usual frame of reference loses its meaning. Terry Pratchett's "Discworld" comedies, set on a rotating disc, use the directions turnwise, widdershins, hubwards and rimwards. On board a Zeppelin airship, which constantly changes its course, the cockpit has no fixed compass bearing from the passenger cabin: it is not very naturally "north". In zero gravity, there is no up or down. Mars does not have a magnetic core, so a compass doesn't work there.

New directions must always be created in opposing pairs, and each **must** be declared with a clear simple sentence of the form "X is a direction." For instance:

Turnwise is a direction. The opposite of turnwise is widdershins.  
Widdershins is a direction. The opposite of widdershins is turnwise.  
Hubwards is a direction. The opposite of hubwards is rimwards.  
Rimwards is a direction. The opposite of rimwards is hubwards.

It is then possible to write, say, that:

Ankh-Morpork is hubwards of Lancre and turnwise from Borogravia.

Of course the Map page of the Index for the project normally draws a map based on compass bearings, so it will get a little befuddled by this. But the map drawn in the Index can be given hints to improve its legibility. More on this later, but for now note that










Index map with turnwise mapped as east.

maps turnwise directions as if they were east, that is, pointing rightwards on the page. (This has no effect on the story file produced; it does not mean turnwise is simply a new name for east; it affects only the look of the Index map, which is only a convenience for the author in any case.)




At one time, directions had to have shortish names (up to three words only), but that's no longer true:

Just the tiniest smidge off magnetic north is a direction. The opposite of just the tiniest smidge off magnetic north is just the tiniest smidge off magnetic south.

Just the tiniest smidge off magnetic south is a direction. The opposite of just the tiniest smidge off magnetic south is just the tiniest smidge off magnetic north.

- 
-  Start of Chapter 3: Things
  -  Back to §3.25. The location of something
  -  Onward to Chapter 4: Kinds: §4.1. New kinds
  -  Example 40:  **Prisoner's Dilemma** A button that causes a previously non-existent exit to come into being.
  -  Example 41:  **The World of Charles S. Roberts** Replacing the ordinary compass bearings with a set of six directions to impose a hexagonal rather than square grid on the landscape.
  -  Example 42:  **Fore** Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel.
- 

## Examples from Chapter 3: Things

-  Start of this chapter
-  Chapter 4: Kinds
-  Indexes of the examples

2

### Example Bic

Testing to make sure that all objects have been given descriptions.

RB

It may occasionally be useful to check whether all objects in our game have a given property. Here we have a "not for release" section that will run at the start of the game and alert us to any objects lacking description:

"Bic"

Section 1 - Testing descriptions - Not for release

When play begins (this is the run property checks at the start of play rule):  
repeat with item running through things:

```
if description of the item is "":
    say "[item] has no description."
```

## Section 2 - Story

The Staff Break Room is a room.

The player carries an orange, a Bic pen, and a napkin. The description of the orange is "It's a small hard pinch-skinned thing from the lunch room, probably with lots of pips and no juice."

The description of the napkin is "Slightly crumpled."

3

### Example Verbosity 1

RB

Making rooms give brief room descriptions when revisited.

By default, the description of a room is printed every time the player enters a room.

On a device with very limited screen space, however, we might wish to supplant that behavior with "brief" descriptions. In Brief mode, Inform prints room descriptions only when the player enters that room for the first time. Afterwards, the text is skipped, for brevity, though the player can see it again at any time by typing LOOK.

As we saw in the previous chapter, we can set "use options" to control certain aspects of the player's experience. One of the use options is the option to

Use brief room descriptions.

which changes the defaults so that the description of a room is printed only the first time the player enters.

"Verbosity"

Use brief room descriptions.

The Wilkie Memorial Research Wing is a room. "The research wing was built onto the science building in 1967, when the college's finances were good but its aesthetic standards at a local minimum. A dull brown corridor recedes both north and south; drab olive doors open onto the laboratories of individual faculty members. The twitchy fluorescent lighting makes the whole thing flicker, as though it might wink out of existence at any moment.

The Men's Restroom is immediately west of this point."

The Men's Restroom is west of the Research Wing. "Well, yes, you really shouldn't be in here. But the nearest women's room is on the other side of the building, and at this hour you have the labs mostly to yourself. All the same, you try not to read any of the things scrawled over the urinals which might have been intended in confidence."

Test me with "west / east".

If we type "test me" during play, these commands will be carried out automatically, and we can see that when we return to the Research Wing, the description is not given a second time.

Some notes: the player can also turn full-length descriptions on or off with the commands "verbose" and "brief", or set a minimal-description setting with the command "superbrief". This power still belongs to the player even if we have set the use option to show brief room descriptions by default.

Moreover, we can ourselves check what the state of the descriptions is, with

if set to sometimes abbreviated room descriptions: ...  
if set to unabbreviated room descriptions: ...  
if set to abbreviated room descriptions: ...

Finally, it is possible to exercise more precise control over what the player sees on his first and subsequent visits to a room; see the next example for details.

---

#### 4 Example Slightly Wrong

RB

A room whose description changes slightly after our first visit there.

A fairly common effect in interactive fiction is a room which is described differently on the first visit than on subsequent visits. We can produce this effect as follows:

"Slightly Wrong"

Awning is a room. "A tan awning is stretched on tent poles over the dig-site, providing a little shade to the workers here; you are at the bottom of a square twenty feet on a side, marked out with pegs and lines of string. Uncovered in the south face of this square is an awkward opening into the earth."

Slightly Wrong Chamber is south of the Awning. "[if unvisited]When you first step into the room, you are bothered by the sense that something is not quite right: perhaps the lighting, perhaps the angle of the walls. [end if]A mural on the far wall depicts a woman with a staff, tipped with a pine-cone. She appears to be watching you."

Test me with "look / s / look".

Note the "[if unvisited]..." in the description of the Slightly Wrong Chamber. A room is considered to be "unvisited" until after the player has seen its description for the first time.

The bracketed text creates a special rule for printing; we will learn more about these in the sections on text with variations and text with substitutions.

Some further fine print: we might write our condition as "if unvisited", "if the location is unvisited", or "if the Chamber is unvisited" -- all of these constructions

would be acceptable, but in the absence of more specifics, the condition is understood to apply to the object whose description it is.

5

### Example Port Royal 1

RB

A partial implementation of Port Royal, Jamaica, set before the earthquake of 1692 demolished large portions of the city.

"1691"

Fort James is a room. "The enclosure of Fort James is a large, roughly hexagonal court walled with heavy stone. The walls face the entrance to Port Royal Harbour, and the battery of guns is prepared to destroy any enemy ship arriving."

Unless we arrange otherwise, this will be the first room in the game because it is the first we have defined.

For subsequent rooms, we do not have to say explicitly that they are rooms, as long as they are connected to a room on the map. For instance, this will automatically make Thames Street End a room:

Thames Street End is south of Fort James. "The ill-named Thames Street runs from here -- at the point of the peninsula -- all the way east among houses and shops, through the Fish Market, edging by the round front of Fort Carlisle, to the point where the town stops and there is only sandy spit beyond. Lime Street, wider and healthier but not as rich, runs directly south, and to the north the road opens up into the courtyard of Fort James."

Water Lane is east of Thames Street End. "Here Thames Street -- never very straight -- goes steeply southeast for a portion before continuing more directly to the east."

Water Lane runs south toward Queen Street, and facing onto it is the New Prison -- which, in the way of these things, is neither. It did serve in that capacity for a time, and in a measure of the villainy which has been usual in Port Royal from its earliest days, it is nearly the largest building in the town."

If we have some concern that the room name will be confused with an existing name, we can be more explicit about it using "called":

East of Water Lane is a room called Thames Street at the Wherry Bridge. Thames Street at the Wherry Bridge has the description "To the southwest is the fishmarket; directly across the street is the entrance to a private alley through a brick archway."

The Private Alley is south of Thames Street at the Wherry Bridge. "You're just outside the tavern the Feathers. To the north, under a pretty little archway, is the active mayhem of Thames Street, but the alley narrows down to a dead end a little distance to the south."



And now we get "inside", which generates a space treated as its own area on the map.

The Feathers is inside from the Private Alley. "Newly built with brick, replacing the older Feathers tavern that used to stand here. It sells wines in quantity, as well as serving them directly, and the goods are always of the best quality. There's a room upstairs for those wanting to stay the night." The Feathers Bedroom is above the Feathers.

And if we like we can declare a number of rooms for which we will come back and write the descriptions later. There is no obligation for the description to occur at the first definition of the room.

Lime Street is south of Thames Street End.

For efficiency, we can also write multiple sets of connections at once:

Queen Street East is east of Queen Street Middle and south of Private Alley.

Clicking Go will translate this description into a sketchy but working simulation of Port Royal, in which we can type movement commands like EAST or SOUTH to explore the streets. Looking at the World tab of the Index, we can also see a schematic map of the simulation as it currently stands. Like the rest of the Index, this is provided entirely for the author's benefit, and is not visible to the player. (Though if we do decide that we want players to have access to a printed map while they play, Inform can help: we will return to the layout of Port Royal in the chapter on Publishing.)

The following Test command allows us to type TEST ME and explore the map we just devised:

Test me with "s / e / e / s / in".

---

6



### Example Up and Up

RB

Adding a short message as the player approaches a room, before the room description itself appears.

Sometimes when a player moves from one room to another, we want to imply that a considerable amount of time elapses, or that something interesting occurs on the way. In that case, we might want to print more than just the room description itself. Here is how we might define a couple of rooms that are far apart:

"Up and Up"

The Plain of the Skull is below the Endless Tower. The description of the Plain of the Skull is "A vast and trackless plain, enlivened only by the bones of those who have previously tried and failed to cross. Above you is the Endless Tower, which rises half-way to the moon."

The description of the Endless Tower is "From up here the Plain of the Skull seems only a small bald patch: the world is round and most of it is covered with trees. Far off to the southwest is a shimmering surface that might be water; but there are no signs of cities or civilizations, only the lizard-skeletons."

And now we borrow from the instructions on Actions to create our actual message. "Before..." introduces a rule that occurs when the player tries to do something; in this case, we will make a Before rule for going to the tower.

Before going to the Endless Tower:

say "You climb... and climb... and climb... The sun sets. The moon rises. The wind begins to blow. You continue to climb..."

The player carries a bit of harness. The description of the harness is "A strip of worked leather and a loop of metal, scavenged from one of the skeletons on the plain. Without it, you might think your entire quest was in vain."

Test me with "look / up".

---

7



### Example Starry Void

RB

Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.

Sometimes we may want a room to be visible from the outside in one location, but treated as a separate location when we are inside. The simplest way to do this is to make the exterior form of the object into a door object, and to describe it differently from different vantage points. (Doors in general are described more fully in the Doors section of the Things chapter.)

"Starry Void"

The Center Ring is a room.

The magician's booth is a door. "[if the player is in Center Ring]A magician's booth stands in the corner, painted dark blue with glittering gold stars.[otherwise if the magician's booth is closed]A crack of light indicates the way back out to the center ring.[otherwise]The door stands open to the outside.[end if]".

Here we've arranged for the booth to be described in the initial room description in different ways depending on where the player is when viewing it. We might like to do the same if the player takes a closer look:

Instead of examining the magician's booth in the Center Ring:

say "It is dark blue and glittering with gold stars. [if the booth is open]The door currently stands open[otherwise]It has been firmly shut[end if]."

Instead of examining the magician's booth in the Starry Void:

say "The booth door is [if the magician's booth is open]wide open[otherwise]shut, admitting only a thin crack of light[end if]."

And now we put it in place:

The magician's booth is inside from Center Ring and outside from Starry Void.

...and make sure that the booth-and-door object responds to all the names we have used for it in different places:

Understand "door" or "of" or "the" or "light" or "crack" or "thin crack" as the booth.

Test me with "examine booth / open door of the booth / in / examine door / close door / look / examine crack of light".

A final nice touch, if we're so inclined, is to borrow from the Basic Actions chapter and make the player automatically open the booth door before trying to enter:

Before going through the closed magician's booth:  
say "(first opening the door of the booth)[command clarification break]";  
silently try opening the booth.

For the contrasting case of a space that is nested inside another place and is not its own room -- say a stall at an open-air market, or a rowboat on a lake -- see the example "Tamed".

---

8

## ★ Example Port Royal 2

RB

Another part of Port Royal, with less typical map connections.

"1691"

Thames Street End is a room.

If we check out a map of historic Port Royal, we find that Thames Street End bends around the northwest tip of the peninsula and becomes the (very) roughly north/south Fisher's Row. We can't put Fisher's Row south of Thames Street End, though, because Lime Street is already going that way. So instead, let's have a map connection that bends around from west to north:

West of Thames Street End is north of Fisher's Row.

Now continuing west along Thames Street, or north along Fisher's Row, will bring us around the corner in question. Asymmetric map connections should be used carefully. They're good for representing the layout of the real world, which tends not to be laid out on a convenient square matrix, but if exits are not described clearly they can be disorienting for the player. So let's be sure to make things clear:

The description of Fisher's Row is "A waterfront street that runs south towards Chocolata Hole, where the small craft are harboured. It also continues north around the tip of the peninsula from here, turning into the east-west Thames Street."

Meanwhile, suppose Fort James is in a prominent position, raised a bit from its surroundings; maybe the player should be able to go down from there, as well as south, to get to Thames Street End.

Thames Street End is down from Fort James. Thames Street End is south from Fort James.

But we don't want the upward direction to work:

Up from Thames Street End is nowhere.

Test me with "n / d / u / w / e / n / s".

---

9  **Example The Unbuttoned Elevator Affair**

RB

A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.

This is very simple. The interior of the elevator is a single room, but which is mapped east of both of its termini. The reverse map connection, west from the elevator, can only go to a single room, and that's what determines which floor the elevator is on.

"The Unbuttoned Elevator Affair"

UNCLE Headquarters is a room. "The steel nerve-center of the free world's battle against the Technological Hierarchy for the Removal of Undesirables and the Subjugation of Humanity. Being against technology, we have only a very simple elevator to the east."

Del Floria's Tailor Shop is a room. "Only trained anti-THRUSH agents recognise the booth in the east wall as a secret elevator."

The Secret Elevator is east of UNCLE Headquarters. The Secret Elevator is east of Del Floria's Tailor Shop.

After going to the Secret Elevator:

say "The doors automatically close, there is a rush of motion, and they open again.";

if UNCLE Headquarters is mapped west of the Secret Elevator, now Del Floria's Tailor Shop is mapped west of the Secret Elevator;

otherwise now UNCLE Headquarters is mapped west of the Secret Elevator; continue the action.

Test me with "east / west / east / west".

---

10

 **Example Port Royal 3**  
Division of Port Royal into regions.

RB

"1691"

We should go ahead and do all our room definitions first...

Fort James is a room. "The enclosure of Fort James is a large, roughly hexagonal court walled with heavy stone. The walls face the entrance to Port Royal Harbour, and the battery of guns is prepared to destroy any enemy ship arriving."

Thames Street End is south of Fort James. "The ill-named Thames Street runs from here -- at the point of the peninsula -- all the way east among houses and shops, through the Fish Market, edging by the round front of Fort Carlisle, to the point where the town stops and there is only sandy spit beyond. Most of that stretch is full of people at all hours. Imported goods are moved off of ships and taken to distributors; exported goods are brought to be loaded; and there is one public house and brothel for every ten inhabitants.

Lime Street, wider and healthier but not as rich, runs directly south, and to the north the road opens up into the courtyard of Fort James."

Lime Street is south of Thames Street End. West of Thames Street End is north of Fisher's Row. The description of Fisher's Row is "A waterfront street that runs south towards Chocolata Hole, where the small craft are harboured. It also continues north around the tip of the peninsula from here, turning into the east-west Thames Street."

Thames Street End is down from Fort James. Up from Thames Street End is nowhere.

Water Lane is east of Thames Street End. "Here Thames Street -- never very straight -- goes steeply southeast for a portion before continuing more directly to the east.

Water Lane runs south toward Queen Street, and facing onto it is the New Prison -- which, in the way of these things, is neither. It did serve in that capacity for a time, and in a measure of the villainy which has been usual in Port Royal from its earliest days, it is nearly the largest building in the town."

East of Water Lane is a room called Thames Street at the Wherry Bridge. Thames Street at the Wherry Bridge has the description "To the southwest is the fishmarket; directly across the street is the entrance to a private alley through a brick archway."

The Fishmarket is southwest of Thames Street at the Wherry Bridge.

The Private Alley is south of Thames Street at the Wherry Bridge. "You're just outside the tavern the Feathers. To the north, under a pretty little archway, is the active mayhem of Thames Street, but the alley narrows down to a dead end a little distance to the south."

The Feathers is inside from the Private Alley. "Newly built with brick, replacing the older Feathers tavern that used to stand here. It sells wines in quantity, as well as serving them directly, and the goods are always of the best quality.

There's a room upstairs for those wanting to stay the night." The Feathers Bedroom is above the Feathers.

Thames Street by the King's House is east of Thames Street at the Wherry Bridge. "The King's House is reserved for the use of the Governor, but he does not live in it, and it is frequently being rented out to some merchant so that the government will at least derive some value from it. It is nearly the least interesting establishment on Thames Street, and the crowd -- which, to the west, is extremely dense -- here thins out a bit."

Thames Street before Fort Carlisle is east of Thames Street by the King's House. "Here Thames Street, formerly a respectable width, narrows to a footpath in order to edge around the front of Fort Carlisle, underneath the mouths of the cannon.

There are no buildings on the harbour side of Thames Street at this point, which means that you have an unusually good view of the ships at dock, water beyond, and the Blue Mountains rising on the other side of the harbour."

South of Thames Street before Fort Carlisle is a room called Fort Carlisle. The description of Fort Carlisle is "Handsomely arrayed with cannons which you could fire at any moment -- though of course there are ships at dock which might be in the way."

Queen Street End is south of Lime Street.

Queen Street Middle is east of Queen Street End.

Queen Street East is east of Queen Street Middle and south of Private Alley.

Queen Street at the Prison is east of Queen Street East.

Now, if we like, we can create regions to distinguish the coast from the portions of town that aren't on the water:

Inland is a region. Queen Street End, Queen Street Middle, Queen Street East, Private Alley, Lime Street, and Queen Street at the Prison are in Inland.

Waterfront is a region. Thames Street before Fort Carlisle, Thames Street by the King's House, Thames Street at the Wherry Bridge, Water Lane, Fishmarket, Fisher's Row, and Thames Street End are in Waterfront.

There's no rule that regions must be contiguous, so we could if we like make a region consisting just of the two forts:

Military Holdings is a region. Fort Carlisle and Fort James are in Military Holdings.

And we might make the Feathers Tavern part of the Inland area, but within its own subcategory:

Tavern is a region. It is in Inland. Feathers and Feathers Bedroom are in Tavern.

Now the index map will be colored to reflect our regions, and later in the game development we would be able to make rules that affect just one region at a time.

11

### Example First Name Basis

RB

Allowing the player to use different synonyms to refer to something.

Sometimes we create objects that we want the player to be able to call by different names: a television that should also answer to "tv" and "telly", for instance, or a refrigerator the player might also call "fridge". In this case, we can use instructions like

```
Understand "tv" and "telly" as the television.
```

to add extra names to the object we've defined.

```
"First Name Basis"
```

```
The Crew Lounge is a room. "Deliberately spartan: the crew feels weight restrictions here first, so there aren't any chairs, just a few thin pads on the ground."
```

```
The holographic projector is a device in the Crew Lounge. "The one major source of entertainment is the holographic projector, a top of the line Mithon 9000, on which you view every beam you can get." Understand "holo" or "holograph" or "Mithon" or "9000" as the projector.
```

```
The description of the projector is "[if switched on]The projector is now playing a documentary about the early politics of the Mars colony.[otherwise]The air above the projector is disappointingly clear.[end if]".
```

(This description is for local color; we will learn more about devices, and conditions like "if switched on", later in this chapter.)

By default, Inform does not understand the names of an object's kind as referring to that object, unless the object has no other name of its own. We can change this, if we like, by defining names that should be applied to everything of a given kind:

```
Lewis and Harper are men in the Crew Lounge. Understand "man" or "guy" or "chap" or "lad" or "male" as a man. Understand "men" or "chaps" or "lads" or "guys" or "males" as the plural of a man.
```

```
The description of Lewis is "A wiry, excitable engineer who just signed aboard last week." The description of Harper is "Harper's a good guy: taciturn when sober, affectionate when drunk, but rarely annoying in either state."
```

```
Test me with "x holo / x man / lewis / x guy / harper / turn on projector / x holo projector / get men".
```

Inform's naming abilities go considerably further, in fact: we can also instruct it to understand words only under certain circumstances, or only when they appear with other words. Fuller details may be found in the chapter on Understanding.

---

12

### ★ Example **Midsummer Day**

RB

A few sentences laying out a garden together with some things which might be found in it.

"Midsummer Day"

East of the Garden is the Gazebo. Above is the Treehouse. A billiards table is in the Gazebo. On it is a trophy cup. A starting pistol is in the cup. In the Treehouse is a container called a cardboard box.

Test me with "up / x box / d / e / x table / x cup / x pistol / get cup".

---

13

### ★ Example **Tamed**

RB

Examples of a container and a supporter that can be entered, as well as nested rooms.

Within a room, we might have containers and supporters that a player can enter. A chair, stool, table, dais, or pedestal would be an enterable supporter (anything we would describe a person as being "on"); a cage, hammock, or booth would be an enterable container (because we would describe the person as being "inside").

When the player is in or on something, he is able to see the rest of the contents of the room, but a note such as "(in the hammock)" or "(on the poster bed)" is added to the room title when he looks around.

Here is an example to show off the possibilities:

"Tamed"

The Center Ring is a room. The cage is in the Center Ring. A lion is an animal in the cage. The cage is enterable, openable, transparent, and closed.

Notice that we made the cage transparent. Strictly speaking it is not made of transparent materials, but we can see into (or out of) a closed cage due to the gaps between the bars, so that from Inform's point of view a cage behaves much like a large sturdy glass box. (If we really wanted to make a distinction between, say, an airtight container and one with perforations, we could do so, but Inform does not model such nuances by default.) If a container is not transparent, we can see into and out of it only when it is open.



Supporters are a bit more straightforward because there is no circumstance in which they separate the player from the rest of the world:

The pedestal is in the Center Ring. It is enterable.

And in fact we can tell Inform that the player starts on the pedestal with this line:

The player is on a pedestal.

Now the player will begin there rather than just in the Center Ring.

This last bit is an entirely unnecessary bit of local color, but if we're going to keep getting into and out of the lion's cage, we ought to expect him to take notice:

Every turn when the player is in the cage:  
if a random chance of 1 in 2 succeeds, say "The lion eyes you with obvious discontent."  
otherwise say "Though the lion does not move, you are aware that it is watching you closely."

Randomness is explained more completely in the chapter on Change, and every turn rules in the chapter on Time.

Finally, we might want a container whose interior is modeled as its own separate room: say, a magician's booth in which volunteers are made to disappear.

The magician's booth is a container in Center Ring. "Off to one side is a magician's booth, used in disappearing acts. The exterior is covered with painted gilt stars." The booth is enterable, open, not openable, and fixed in place.

Now we create our other location:

Inside from the Center Ring is the Starry Vastness.

...which handles the case of the player typing >IN. (We will not assume by default that he wants to get into the cage with the lion, this being obviously perilous.) But we also want to make sure that the player who types >ENTER BOOTH winds up in the same place, so we should add:

Instead of entering the magician's booth: try going inside.

Test me with "get in cage / open cage / get in cage / z / close cage / out / open cage / get on pedestal / get off / look / enter booth / out".

To begin with the title:

## "Disenchantment Bay"

There are many Disenchantment Bays across the world, named by eighteenth-century ships' captains - one in Antarctica, another in Tasmania, for instance. The most famous is probably the one where Lewis and Clark's expedition broke through to the Pacific. But ours is the one in Alaska, named in 1791 by a Spanish navigator who had hoped it might lead to the fabled Northwest Passage, and all of this history is beside the point since the game is set in the present day.

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions, which can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

We might want to start with the glass case.

The Cabin contains a glass case. In the glass case is a collection of fishing rods.

Now Inform will have guessed that the case is a container, but its default idea of a container is something like a bucket: permanently open and not able to be opened and shut. We can change that:

The case is closed, transparent, and openable.

We get a similar set of guesses if we write

The bench is in the cabin. On the bench are some blue vinyl cushions.

Using "some" rather than "a" or "the" tells Inform that the cushions are to be referred to as a plural object in the future. And because of the "on the bench..." phrase, Inform will guess that the bench is a supporter and that it is fixed in place and cannot be moved from room to room. We do have to tell it that the bench can be sat on, though:

The bench is enterable.

And now a short script, so that if we type TEST ME, we experiment with the case and bench:

Test me with "examine case / get rods / open case / get rods / sit on bench / take cushions / get up"

If we compile our last version of the cabin, we get a room where the glass case and the bench are listed separately from the room description, even though they have already been mentioned once. We can prevent this by making the already-mentioned things scenery:

"Disenchantment Bay"

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions, which can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, and openable. The case is scenery.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.

Generally speaking, it is a good idea to recognize the player's attempts to interact with any objects mentioned in the room description, so we should also provide

Some navigational instruments, some scratched windows, a sign, a radar display, and some radios are scenery in the cabin.

Test me with "examine instruments / x windows / x sign / x display / x radios".

The door and the view will need to be done as well, but they are special cases which we will get to shortly.

As noted, making something scenery also means that the player will be prevented from picking it up and carrying it away. This is sensible, though: if an object can be removed from the room where it first appears, we should be careful about mentioning it in the main room description; otherwise, it will continue to be described as present even when someone has carried it off.

---

16

### ★ Example Replanting

RB

Changing the response when the player tries to take something that is scenery.

By default, "TAKE OAK" in the example above will produce the response "That's hardly portable." This is fine under many circumstances, but also a bit generic, so we might want to override it for a specific game.

"Replanting"

The Orchard is a room. "Within this quadrille of pear trees, a single gnarled old oak remains as a memory of centuries past." The gnarled old oak tree is scenery in the Orchard.

Instead of taking some scenery: say "You lack the hulk-like strength."

Test me with "take oak".

Here we've used an "instead" rule; we will learn more about these in the section on actions. This allows us to define our own results for taking an object.

Note: "scenery" is a property of an object (about which we will hear more later). So when we use it in rules, we can talk about "some scenery", "something that is scenery", or even "a scenery thing" -- the last one doesn't sound much like English, but is a more plausible construction with other adjectives.

---

17

★ **Example Disenchantment Bay 3**

RB

Disenchantment Bay: adding a view of the glacier.

Suppose we wanted to have the glacier visible from the Cabin of our boat, and anywhere else we might add to the game:

The view of the Malaspina glacier is a backdrop. It is everywhere. The description is "The Malaspina glacier covers much of the nearby slope, and -- beyond it -- an area as large as Rhode Island."

---

18

★ **Example Disenchantment Bay 4**

RB

Disenchantment Bay: fleshing out the descriptions of things on the boat.

Currently we have provided objects for most of what is on the boat, but it's not very interesting to look at. We might want to give some more description to these things.

"Disenchantment Bay"

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions, which can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, and openable. The case is scenery.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.

Some navigational instruments, some scratched windows, a radar display, and some radios are scenery in the cabin.

The description of the instruments is "Knowing what they do is the Captain's job."

The description of the windows is "They're a bit the worse for wear, but you can still get an impressive view of the glacier through them. There were whales earlier, but they're gone now."

The description of the radar is "Apparently necessary to avoid the larger icebergs."

The description of the radios is "With any luck you will not need to radio for help, but it is reassuring that these things are here."

The order in which we define these things is fairly open. We could also define an object so:

A sign is scenery in the Cabin. The description is "You can get half-day and full-day sight-seeing tours, and half-day and full-day fishing trips."

Where "the description" is assumed to refer to the thing most recently defined, if no object is specified.

The view of the Malaspina glacier is a backdrop. It is everywhere. The description is "The Malaspina glacier covers much of the nearby slope, and -- beyond it -- an area as large as Rhode Island."

Test me with "examine sign / examine glacier / examine instruments / examine windows / examine radar / examine radios / take the cushions / take the glacier".

These last two commands show how scenery and backdrops are automatically impossible for the player to take.



Some general advice about creating objects with unusual or awkward names, and a discussion of the use of printed names.

Occasionally it is useful to give something a printed name because we want to call it something extremely long-winded; give one thing a name that is the subset of the name of something else; or use words such as "with" or "and" that are likely to confuse Inform into thinking that the object name ends before it actually does.

Often it is enough to preface these ambiguously-titled things with "a thing called..." or "a supporter called..." or the like, as here:

South of Spring Rolls is a room called Hot and Sour Soup.

prevents Inform from trying to read "Hot and Sour Soup" as two separate rooms, while

The player carries an orange ticket. The player carries a thing called an orange.

creates two objects instead of the one orange ticket that would result if the second sentence were merely "The player carries an orange."

Really long names can be a bit cumbersome. For example:

The player carries a thing called an incriminating photograph of a woman with blonde hair.

So we might instead give the photograph a printed name:

"Laura"

The City of Angels is a room. The incriminating photograph is carried by the player. The printed name of the incriminating photograph is "incriminating photograph of a woman with blonde hair".

Now we've gotten around any awkwardness with printing the name -- but we also need to understand when the player refers to the photograph. When we define the names of objects under normal circumstances, Inform takes care of this automatically, but if we have especially set the printed name, we must also specially define the appropriate terms for the player to use. For this we need "understand", which will be explained in much more depth in a later chapter:

Understand "woman" or "with" or "blonde" or "hair" or "of" or "a" as the incriminating photograph.

Test one with "x photograph / x incriminating photograph of a woman with blonde hair / x hair / x blonde / x woman with blonde hair / x incriminating photograph of a woman".

That's probably as far as we really need to go, and if you are satisfied with this behavior, there is no need to read on.

One possible objection to this solution is that Inform will accept some nonsensical formulations as applying to the photograph: for instance, it will allow >EXAMINE PHOTOGRAPH OF, >X BLONDE PHOTOGRAPH WOMAN INCRIMINATING, or even >X OF ...though in the case there were two items with "of" names, the game would disambiguate with a question such as "Which do you mean, the incriminating photograph of a woman with blonde hair or the essence of wormwood?"

Traditionally, Inform has tended to be fairly flexible about word order, preferring to err in the direction of leniency. On the other hand, there are times when we need

more exacting rules in order to distinguish otherwise similar cases.

Two features allow us to specify more exactly if we so desire. The first is that, if we specify a whole phrase as the name of something, all the words in that phrase are required, in the order given. Thus "Understand "blonde hair" as the photograph" would require that both "blonde" and "hair" be present, and would not recognize >X BLONDE, >X HAIR BLONDE, or >X HAIR.

Second, we can create tokens, such as "Understand "blonde hair" or "hair" as "[hair]", and then use these tokens in match phrases. This saves a good deal of time when we want to specify a number of different but fussy alternatives. So, for instance, here is a drawing that would not respond to >X OF, or >X BROWN EYES, but would respond to >X DRAWING OF MAN WITH BROWN EYES, >X MAN WITH BROWN EYES, and so on:

The drawing is carried by the player. The printed name of the drawing is "drawing of a man with brown eyes".

Understand "eyes" or "brown eyes" as "[brown eyes]". Understand "man" or "man with [brown eyes]" or "brown-eyed man" as "[man]". Understand "[man]" or "drawing of [man]" or "drawing of a [man]" as the drawing.

Test me with "test one / test two".

Test two with "x drawing / x man / x of / x drawing of man / x drawing of a man / x drawing of a man with brown eyes / x drawing of a brown-eyed man / x brown eyes".

Further refinements are possible: the "privately-named" attribute tells Inform not to try to understand the source name of an object at all, so if we write

The purple rabbit is a privately-named thing.

...the player will not be able to refer to it as "purple" or "rabbit" or "purple rabbit".

There are also ways to make names to refer to entire kinds of objects (so "dude" will refer to any man in the game); to specify names that only refer to objects in the plural (so GET PICTURES will pick up several pictures together); to reflect an object's properties (so "red apple" works only as long as the apple is in fact red); or even to refer to the object's relationships to other objects (so "bottle of wine" works only when wine is indeed in the bottle). All these refinements are discussed in the chapter on Understanding.

We mentioned that there is a door out to the deck in our example. The following two sentences will create both the door and the other room:

The cabin door is south of the Cabin and north of the Deck. It is a door and scenery.

Now Inform has constructed a generic room called "Deck" to the south. It has neither a description nor any contents yet, but we could fix that in time. It does have a view of the glacier, though, since we defined the glacier view to be everywhere.

21



## Example Escape

RB

Window that can be climbed through or looked through.

Suppose we want to offer the player a window he can climb through, instead of a boring ordinary door. Our window will be like a door in that it connects two rooms, appears in both places, and impedes movement when it is shut. But we also want to add that we can look through it and see what lies on the other side; and we further want to understand "climb through window" or "jump through window" as attempts to pass through it.

We'll start by defining a couple of rooms and making the window a door between them.

"Escape"

Your Bedroom is a room. The bedroom window is a door. It is west of Your Bedroom and east of the Grassy Slope.

Now we have a "bedroom window" object which can be entered. Now, to catch the case where the player types "LOOK THROUGH WINDOW":

Instead of searching the window:

say "Through the window, you make out [the other side of the window]."

The other side of a door is always defined to be the room that we are not currently in when doing the check. When we are in the bedroom, the other side will be the grassy slope, and vice versa. "Searching" is the action that occurs when the player attempts to LOOK THROUGH something. (To review what grammar gives rise to what actions, we can always consult the Actions portion of the Index.)

Next we want to cover the case where we climb through the window:

Instead of climbing the window:

try entering the window.

And because "climb window" is understood but "climb THROUGH window" is not, we will have to borrow from the chapter on Understanding to add some new vocabulary to the game (and we'll add Jump too, while we're at it):

Understand "climb through [something]" as climbing. Understand "jump through [something]" as climbing.



Now the final piece: Inform will already keep the player from going through a closed window, but it will say "You can't, since the bedroom window is in the way." This is probably not ideal, so we can replace the instruction thus:

Instead of going through the closed window:  
say "The window is shut: you'd break the glass."

Test me with "look through window / climb through window / open window / climb through window / look through window / close window / e / open window / e".

---

22



### Example Garibaldi 1

RB

Providing a security readout device by which the player can check on the status of all doors in the game.

Suppose we would like to allow the player to view the status of all the doors functioning in the game; and we want to identify those doors by mentioning which two rooms they connect. The following uses some techniques that will be covered in later chapters, but the basic idea may be obvious:

"Garibaldi"

The security readout is a device. The description of the readout is "The screen is blank."

Instead of examining the switched on security readout:

say "The screen reads: [fixed letter spacing]";  
say line break;  
repeat with item running through doors:  
say line break;  
say " [item] ([front side of the item]/[back side of the item]): [if the item is locked]LOCKED[otherwise]UNLOCKED[end if]";  
say variable letter spacing;  
say paragraph break.

It is more or less arbitrary which room winds up as the "front side" and which as the "back", but in this case it hardly matters.

The player carries the security readout.

The Docking Bay is a room. The inner airlock is a door. It is north of the Docking Bay and south of the Zocalo. The inner airlock is lockable and unlocked. The outer airlock is lockable and locked. It is a door. It is south of the Docking Bay and north of Space.

The quarantine seal is a door. It is west of the Zocalo and east of Medlab. Quarantine seal is locked.

The security pass unlocks the inner airlock. The player carries the security pass.

Test me with "x readout / turn on readout / x readout / lock inner airlock with security pass / x readout".

23

### Example Disenchantment Bay 6

RB

Disenchantment Bay: locking up the charter boat's fishing rods.

It stands to reason that the captain wouldn't let just anyone meddle with his fishing equipment; maybe he keeps that case locked. We could replace the case description with this one, instead:

The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, openable, lockable, and locked. The case is scenery. The small silver key unlocks the case.

Now there's a silver key that will unlock it -- though since we haven't said where the key is, the player will never be able to find it in the game. (If we look at the World index, we find "small silver key" right at the bottom, not inside any of the rooms. That is as good as not existing at all -- though we usually use the term "out of play" - - but as we will later see, it is possible to have things initially out of play but brought into existence later on.)

24

### Example Neighborhood Watch

RB

A locked door that can be locked or unlocked without a key from one side, but not from the other.

Suppose we want a locked door that can be opened with a key, but is also openable by hand without a key from one side only. We start by defining an ordinary lockable door and the key that controls it:

"Neighborhood Watch"

The shabby door is a door. It is outside from the Studio Apartment and inside from the Rickety Stairwell. The shabby door is locked.

The brass key is carried by the player. It unlocks the shabby door.

The next part is going to require that we modify the normal operation of the "lock" command. "Lock" ordinarily requires that the player supply two objects: a thing he wants to unlock, and the key he wants to use on it. The full command is LOCK DOOR WITH THE KEY, and Inform will not accept simply LOCK DOOR as locking.

Therefore, we're going to need to create our own new variant on the lock verb (and the unlock verb, while we're at it). The full procedure for this is laid out in the chapters on Action and Understanding, but here is an example:

Understand "lock [something]" as locking keylessly. Locking keylessly is an action applying to one thing.

Here we've created a new action -- locking something without a key -- and we've told Inform to understand LOCK DOOR as this action, rather than an incomplete command to LOCK DOOR WITH SOMETHING.

Now we add some instructions so that the game will not let us use this keyless unlocking command unless we're in the right place or are properly equipped:

Check locking keylessly:

- if the noun is not a door, say "[The noun] is not something you can lock." instead;
- if the noun is locked, say "[The noun] is already locked." instead;
- if the player carries the brass key and the player is in the Stairwell, try locking the noun with the brass key instead;
- if the player is in the Stairwell, say "You can't lock the door from this side without the key." instead.

This check rule is performed before the keyless locking action succeeds. The first thing we do is try to use the key if the player is outside and has the key: this way, LOCK DOOR will turn automatically into LOCK DOOR WITH THE KEY, under circumstances where that is both possible and necessary.

The second thing is to check whether the player is outside but keyless, and, if so stop the action from being performed successfully. Here we print a failure message followed by the word "instead", which tells Inform that we've substituted some other outcome for the usual performance of the action.

Now we're reasonably sure that the player is only locking keylessly in the case that he is inside the Studio. (We might have to do a more thorough check for this if there were more than two rooms, but as it is, the player can only be in the Stairwell or in the Studio, so if we have ruled out the Stairwell, we are safe.) So now we want to add what happens when locking-without-a-key command succeeds:

Carry out locking keylessly:

- now the noun is locked.

That's it. We've just told Inform to make the door be locked. "Now..." syntax will be explained more thoroughly in the chapter on change. But we still haven't described to the player what just happened, so let's provide a description of that, too:

Report locking keylessly:

- say "You flip over the deadbolt to lock [the noun]."

And now we have to do a similar set of things for unlocking:

Understand "unlock [something]" as unlocking keylessly. Unlocking keylessly is an action applying to one thing.

Check unlocking keylessly:

- if the noun is not a door, say "[The noun] is not something you can lock." instead;

if the noun is unlocked, say "[The noun] is already unlocked." instead;  
if the player carries the brass key and the player is in the Stairwell, try  
unlocking the noun with the brass key instead;  
if the player is in the Stairwell, say "You can't unlock the door from this side  
without the key." instead.

Carry out unlocking keylessly:  
now the noun is unlocked.

Report unlocking keylessly:  
say "You flip over the deadbolt to unlock [the noun]."

Test me with "unlock door / drop key / open door / out / close door / lock door /  
open door / in / get key / out / close door / lock door / unlock door".

Some (but not all) of this work is done for you if you like by the Locksmith  
extension. If you prefer, you can include that extension, then follow the  
documentation in order to implement the remainder of the scenario. Locksmith takes  
care of implementing the additional locking and unlocking actions, and provides  
some other conveniences.

---

25

### **Example Disenchantment Bay 7**

RB

Disenchantment Bay: making the radar and instruments switch on and  
off.

If we would like the player to be able to turn instrumentation on and off, we could  
add a line to this effect:

The radar, the instruments, and the radios are devices.

And since the captain is probably not navigating blind, we might also want to say

The radar and the instruments are switched on.

---

26

### **Example Down Below**

RB

A light switch which makes the room it is in dark or light.

Suppose we want to have a room with a light switch. Turning the switch off makes  
the room go dark; turning it on restores the light. This kind of switch is an obvious  
candidate as a device.

"Down Below"

Terrifying Basement is a room. The light switch is a switched on device in the  
Terrifying Basement. It is fixed in place.

Here we define our light switch, and we also make it start out as "switched on". The Terrifying Basement will also start out lit (as all rooms do, by default, unless we specifically say that they are dark). We further say that it is fixed in place to avoid the ludicrous possibility of the player picking it up and carrying it away.

Next we add some instructions to control how turning the light switch on and off affects the room light. These borrow from later chapters on actions, but the gist may be obvious anyway:

Carry out switching off the light switch: now the Terrifying Basement is dark.

Carry out switching on the light switch: now the Terrifying Basement is lighted.

Inform already has the idea of light and darkness built in; we will see more about this later, and the Phrasebook (in the Index tab) also contains a list of all the adjectives (lighted, dark, etc) which are important to use here.

Speaking of the Index, the Actions tab contains a list of all the grammar that can be used to activate a given command: for instance, the switching action responds to "switch [something]" or "turn on [something]". In this case, we may want to give the player an extra option or two. It would be pretty natural for a player to try >FLIP SWITCH, so let's add that in:

Understand "flip [something switched off]" as switching on. Understand "flip [something switched on]" as switching off. Understand "flip [something]" as switching on.

The nuances of this will be explored in the chapter on Understanding. What is useful to know here is that we have taught Inform to understand that >FLIP LIGHT SWITCH means to turn it on when the switch is already off; if the switch is already on, FLIP SWITCH means to turn the switch off. Depending on the kind of device we are modeling (button? lever? dial?), we might want to write similar lines for commands such as PUSH, PRESS, PULL, TURN, and so on.

Finally, we need to deal with a special case. In general, the player cannot interact with other things in a dark room because he can't see them, but if we adhered strictly to this it would be impossible for him to find the light switch to turn it back on. So we need something from the chapter on Activities to change this:

After deciding the scope of the player when the location is the Terrifying Basement:  
place the light switch in scope.

Upstairs is above the Terrifying Basement.

Test me with "turn off light / look / flip light switch".

Let's say that our protagonist is about to flee . Obviously, he can't make the journey on foot; he needs transportation.

"Peugeot"

Include Rideable Vehicles by Graham Nelson.

The Lot is a room. The ten-speed bike is a rideable vehicle in the Lot.

We make the ten-speed bike a rideable vehicle because we want to say that the player is on it rather than in it. Then our other room:

Cambridge is east of the Lot.

And now we borrow from the Actions chapter to prevent travel without the proper equipment:

Instead of going to Cambridge when the player is not on the ten-speed bike:  
say "It's a long journey to Cambridge: you'll never make it on foot."

After going to Cambridge:  
say "You begin pedalling determinedly.";  
continue the action.

Test me with "e / get on ten-speed bike / e".

---

28



### Example Disenchantment Bay 8

RB

Disenchantment Bay: a pushable chest of ice for the boat.

We probably do not need a vehicle to ride around our boat, but there might be a heavy ice chest that can only be pushed from room to room:

The ice chest is a closed openable container in the Deck. "A very heavy ice chest sits on the ground." It is fixed in place and pushable between rooms. A quantity of ice is in the chest. The description is "Ready and waiting just in case there's any fish needing to be kept cool."

This anticipates a later chapter, but it would probably be a good idea to hint to the player, if he tries to take the ice chest, that there is another way to move it:

Instead of taking the chest: say "It's too heavy to lift, but you might be able to push it, and just inch it over the frame of the door."

Otherwise, attempts to pick it up will just reply with "That's fixed in place."

---



## Example Hover

Letting the player see a modified room description when he's viewing the place from inside a vehicle.

Suppose we want the player to see a modified room description when he's viewing the place from inside a vehicle. There are several conceivable ways of doing this; the example here shows a rather advanced way, but is very flexible and will let us write all sorts of special cases.

"Hover"

Use full-length room descriptions.

Emerald City is a room. "All the buildings are spires and none of them have doors." The Vast Desert is west of Emerald City. "[if the player is in a vehicle]Outside, a[otherwise]A[end if] trackless waste stretches as far as the eye can see in every direction."

The hover-bubble is a vehicle in the Emerald City. "Your hover-bubble awaits." The description is "The hover-bubble is a clear globe-shaped vehicle capable of transporting you anywhere you could walk, but faster." Understand "bubble" as the hover-bubble. The hover-bubble contains a chocolate wrapper and a parking ticket.

Here's the tricky part, which relies on material from the chapters on Activities and Rulebooks:

The container interior rule is listed before the room description body text rule in the carry out looking rules.

This is the container interior rule:

if the actor is the player and the player is in an enterable thing (called current cage), carry out the describing the interior activity with the current cage.

Describing the interior of something is an activity.

Now we've done that, we can write a "rule for describing the interior" of something, which will print whatever we like:

Rule for describing the interior of the hover-bubble:

say "The hover-bubble is transparent, but tints everything outside very faintly lavender."

In fact, as a special refinement, we could even say:

Rule for describing the interior of the hover-bubble when the hover-bubble contains more than one thing:

say "The hover-bubble is transparent, but tints everything outside very faintly lavender. Beside you you can see [a list of other things in the hover-bubble]."

Definition: a thing is other if it is not the player.

Rule for listing nondescript items of the hover-bubble when the player is in the hover-bubble: do nothing.

Test me with "get in bubble / look / west / take all / look / get out / east".

And now anything that's beside us in the vehicle will be described during that first paragraph, rather than later on.

---

30

 **Example Disenchantment Bay 9**

RB

Disenchantment Bay: enter the charter boat's Captain.

Now finally we can put a Captain in the boat:

The Captain is a man in the Cabin. "The captain sits at the wheel, steering the boat and occasionally checking the radar readout."

---

31

 **Example Belfry**

RB

You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.

"Belfry"

The Belfry is a room. A bat is in the Belfry. The bell is in the Belfry. Some woodworm are in the Belfry. A man called William Snelson is in the Belfry. A woman called the sexton's wife is in the Belfry. A man called a bellringer is in the Belfry.

In the Belfry is a man called the vicar. The indefinite article of the vicar is "your local".

Test me with "look".

---

32

 **Example Gopher-wood**

RB

Changing the name of a character in the middle of play, removing the article.

Suppose that we want a character who starts out with a general epithet ("the bearded man") but is later introduced to the player properly ("Japheth"). In that case, we want to be able to tell Inform to stop using an article once the character has been given a proper name. We can do this like so:

"Gopher-wood"



The Ark is a room. A bearded man is in the Ark.

Instead of examining the bearded man for the first time:  
now the printed name of the bearded man is "Japheth";  
now the bearded man is proper-named;  
say "You peer at him a bit more closely and realize that it's Japheth."

Finally, we need to tell Inform to understand the man's name, but only when he's been introduced. For this purpose, we borrow from the chapter on Understanding:

Understand "Japheth" as the bearded man when the bearded man is proper-named.

Test me with "x japheth / x man / look / x japheth".

---

33

★ **Example Disenchantment Bay 10**

RB

Disenchantment Bay: things for the player and the characters to wear and carry.

At this point we can dress both the Captain and the player with some appropriate props:

The captain wears a baseball cap. The description of the cap is "It says, THE WORST DAY FISHING IS BETTER THAN THE BEST DAY WORKING."

The player is carrying a backpack and a bottle of water. The player is wearing a pair of sunglasses. The description of the sunglasses is "The light off the water and the ice does get pretty bright sometimes."

(At present the backpack can't be worn, but see the next version.)

---

34

★ **Example Disenchantment Bay 11**

RB

Disenchantment Bay: making a holdall of the backpack.

If we wanted, we could make the player's backpack infinitely capacious, so:

The backpack is a player's holdall.

...And now whenever the player character is unable to hold everything, he will automatically stow some of his possessions therein.

This is only useful if the player doesn't have infinite carrying capacity himself, so perhaps we also need

The carrying capacity of the player is 3.

Perhaps mercifully, items which are worn are not counted against the player's carrying capacity. We might want to let him take advantage of that, too:

The backpack is wearable.

This capacity system makes a compromise between the realistic and the absurd: on the one hand, it acknowledges that people can't carry an infinite number of items in their hands, while at the same time providing a sack that can.

Many games will have no use for object-juggling of this kind at all; others will want to be much more rigorous about questions of capacity and volume. Fortunately, it is easy to leave the whole business out by assigning no carrying capacity to anything.

35



### Example **Fallout Enclosure**

RB

Adding an enclosure kind that includes both containers and supporters in order to simplify text that would apply to both.

It may not be immediately obvious why we might want to create new intermediate categories of the kinds hierarchy. But there may be times, for instance, where we would like to make an action that applies in the same way to both containers and supporters, but to nothing else in the game. To avoid creating two nearly-identical rules, we would instead roll the two categories together into one, on the principle that duplicating source text is usually a sign of bad design.

So for instance let's say the player is able to zap objects to make them go away, but any contents -- things inside a container or on top of a supporter -- should always be left as residue. Here's one way we might do this:

"Fallout Enclosure"

Section 1 - Procedure

An enclosure is a kind of thing. A container is a kind of enclosure. A supporter is a kind of enclosure.

Understand "zap [something]" as zapping. Zapping is an action applying to one thing. The Zapping action has a list of things called the remnants.

Carry out zapping an enclosure:

if the noun holds something:

now the remnants is the list of things held by the noun;

repeat with N running through the remnants:

move N to the holder of the noun.

Carry out zapping:

now the noun is nowhere.

Report zapping:

say "You zap [the noun], destroying [them][if the remnants is not empty] and

leaving [the remnants with indefinite articles] behind[end if]."

## Section 2 - Scenario

SuperDuperMart is a room. SuperDuperMart contains some shelves and a cash register.

The shelves support a bottle of Buffout and a container of Jet.

The cash register contains some prewar money, a coin purse, and a bottle cap. The coin purse contains a prewar nickel. It is closed.

The cash register is closed and locked.

Test me with "zap shelves / zap buffout / zap register / zap purse".

36



### Example Brown

RB

A red sticky label which can be attached to anything in the game, or removed again.

"Brown"

The Shipping Room is a room. The red sticky label is a thing carried by the player. The description of the red sticky label is "It reads: AIRMAIL[if the label is part of something (called the parent)]. It is stuck to [the parent][end if]."

A black crate is in Shipping. The description is "A boring black crate." The brown crate is a thing in Shipping. The description is "An ordinary brown crate."

After examining something when the label is part of the noun:  
say "A bright red sticky label is attached to [the noun]!"

Here is the essential point: whenever we ATTACH LABEL TO something, it becomes part of that object.

Instead of tying the red sticky label to something:  
now the red sticky label is part of the second noun;  
say "You stick [the label] to [the second noun]."

And of course the label cannot be stuck to itself or to more than one thing at a time.

Before tying the label to something when the label is part of something:  
if the label is part of the second noun:  
say "[The label] is already stuck to [the second noun]." instead;  
otherwise:  
say "(first freeing the label)[line break]";  
silently try taking the label;  
if the label is part of something, stop the action.

Instead of tying the red sticky label to the label:  
say "That would ruin the label entirely."

Instead of taking the label when the label is part of something:  
now the player carries the label;  
say "You peel the label off again."

Much of the rest is just tidying to make sure that the player's commands are redirected into the right syntax.

Instead of tying something to the label:  
try tying the label to the noun.

Instead of putting the label on something:  
try tying the label to the second noun.

Instead of inserting the label into something:  
try tying the label to the second noun.

Understand the commands "stick" or "apply" as "tie".

We could have created a new "sticking" action, but to keep the example short we will use the built-in "tying" action instead, and respond to the command "stick" just as if it were "tie".

Understand "peel [something]" or "peel off [something]" as taking.

Test me with "i / put label on the black crate / look / x black / x label / get the label / apply label to brown crate / look / x brown / peel off label / stick label to label".

---

37



### Example Disenchantment Bay 12

RB

A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained.

"Disenchantment Bay"

Include Locksmith by Emily Short.

Use scoring.

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions[if the compartment is closed], which can be lifted to reveal the storage space underneath[otherwise], one of which is currently lifted to allow access to the storage compartment within[end if]. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. Understand "rod" as the collection. The case is closed, transparent, openable, lockable, and locked. The case is scenery. The small silver key unlocks the case.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.

A storage compartment is an openable closed container. It is part of the bench. Instead of opening the bench, try opening the storage compartment. Instead of closing the bench, try closing the storage compartment. Instead of pushing or pulling or turning the cushions, try opening the storage compartment. Understand "space" as the storage compartment.

Some nets and a Coke are in the compartment. Understand "net" as the nets. The description of the nets is "They must have something to do with fish as well. Really, you're just here for the sights." The nets are a container.

Some navigational instruments, some scratched windows, a radar display, and some radios are scenery in the cabin. The radar, the instruments, and the radios are devices. The radar and the instruments are switched on.

A screen is part of the radar. The description of the screen is "[if the radar is switched on]Phantom lights move across the screen.[otherwise]The screen is dark.[end if]". Instead of doing something other than examining to the screen, say "It's not good for much but looking at."

The Captain is a man in the Cabin. "The captain sits at the wheel, steering the boat and occasionally checking the radar readout." The captain wears a baseball cap. The description of the cap is "It says, THE WORST DAY FISHING IS BETTER THAN THE BEST DAY WORKING." The captain carries the silver key. The description of the captain is "[The captain] is wearing [a list of things worn by the captain][if the captain carries something] and carrying [a list of things carried by the captain][end if]."

The description of the instruments is "Knowing what they do is the Captain's job." Instead of doing something other than examining to the instruments in the presence of the Captain: say "The Captain glares at you. Clearly you are not welcome to do that."

The description of the windows is "They're a bit the worse for wear, but you can still get an impressive view of the glacier through them. There were whales earlier, but they're gone now." Understand "window" as the windows.

The description of the radar is "Apparently necessary to avoid the larger icebergs."

The description of the radios is "With any luck you will not need to radio for help, but it is reassuring that these things are here."

A sign is scenery in the Cabin. The description is "You can get half-day and full-day sight-seeing tours, and half-day and full-day fishing trips."

The view of the Malaspina glacier is a backdrop. It is everywhere. The description is "The Malaspina glacier covers much of the nearby slope, and -- beyond it -- an area as large as Rhode Island." Understand "view of the surrounding bay" or "surrounding bay" as the view.

The cabin door is south of the Cabin and north of the Deck. It is a door and scenery. The description of the Deck is "The whole back half of the boat is open, allowing you to view the surroundings without intervening windows -- if you can stand the cold."

The ice chest is a closed openable container in the Deck. "A very heavy ice chest sits on the ground." It is fixed in place and pushable between rooms.

A quantity of ice is in the Deck. "All around the boat bob chunks of glacier ice." Understand "glacier ice" as the quantity. The description is "Curiously cooled into funny-shaped chunks." The printed name of the quantity is "glacier ice".

Instead of taking the quantity of ice when the player is not carrying the nets:  
if the quantity of ice is handled, continue the action;  
say "You are having a hard time fishing out the ice with your bare hands."

Instead of taking the quantity of ice when the player is carrying the nets:  
if the quantity of ice is handled or the quantity of ice is in the nets, continue the action;  
now the quantity of ice is in the nets;  
say "You scoop up the ice with the net."

Instead of taking the chest: say "It's too heavy to lift, but you might be able to push it, and just inch it over the frame of the door."

The player is carrying a backpack. The player is wearing a pair of sunglasses. The description of the sunglasses is "The light off the water and the ice does get pretty bright sometimes."

The backpack is a player's holdall. The carrying capacity of the player is 3. The backpack is wearable.

Instead of asking the Captain for the key:  
say ""Sure, you can -- well, get me a drink first, would you?""

Instead of asking the Captain for the key when the Captain is carrying a cold Coke and the Captain is carrying the key:  
move the key to the player;  
say ""Here, knock yourself out.""

Instead of asking the Captain for the key when the Captain is not carrying the key: say ""I already gave it to you. You didn't lose it, did you?""

Heat is a kind of value. The heats are cold, cool, room temperature, and warm.

A beverage is a kind of thing. A beverage can be open or closed. A beverage can be openable or unopenable. A beverage is always edible and openable. A beverage has a heat. A beverage is usually warm. The Coke is a beverage. The beer is a beverage. The beer is in the backpack.

Instead of giving or showing a beer to the Captain:

say "'I don't drink on the job, thanks,' he says. 'You can help yourself if you want it, though.'"

Instead of giving or showing a Coke to the Captain:

say "'It needs chilling,' the Captain remarks, disgruntled."

Instead of giving or showing a cold Coke to the Captain:

move the Coke to the Captain;  
increase the score by 2;  
say "'Ah, thank you,' he says. How he can drink an iced soda on a day like today is an open question, but Alaskans are special."

Every turn when the quantity of ice is in the ice chest:

repeat with item running through beverages in the ice chest:  
let the current heat be the heat of the item;  
if the current heat is not cold, now the heat of the item is the heat before the current heat.

Before printing the name of a beverage (called the drink):

say "[heat of the drink]".

Understand the heat property as describing a beverage.

The maximum score is 5.

After taking the fishing rods:

end the story finally;  
increase the score by 3;  
say "Success is yours! (Now if only you knew anything about fishing.)"

Test me with "test first / test second / test third".

Test first with "x captain / open case / i / ask captain for the key / give beer to captain / open bench / x nets / get nets / get coke / give coke to captain".

Test second with "s / open chest / drop nets / get glacier ice / get nets / get glacier ice / g / put glacier ice in chest / get coke / put coke in chest".

Test third with "x coke / g / g / g / get coke / n / give coke to captain / ask captain for key / open case / get rods".



Suppose we want a character who carries hidden objects, but only while he is wearing his jacket. If we deprive him of this, his other possessions become known. Furthermore, if we ever search him, his possessions also become known, and can thereafter be mentioned by us.

"Search and Seizure"

Size is a kind of value. The sizes are small, medium, and large. A thing has a size. A thing is usually small.

A thing can be licit or contraband. A thing is usually licit.

A thing can be discovered or secret. A thing is usually secret.

Now, we want the character to be able to hide small things if he has some sort of concealing garment on. We also want to be able to see anything that the player has already found once, perhaps by using the >SEARCH PERSON command. So:

Rule for deciding the concealed possessions of someone (called the suspect):  
if the particular possession is discovered, no;  
if the suspect wears something and the particular possession is small, yes;  
otherwise no.

The following rule, borrowed from a later chapter, assures that any items that are ever mentioned to the player will be treated as known from then on:

Before printing the name of something (called discovery): now the discovery is discovered.

The Customs House is a room. The smuggler is a man in the Customs House. The smuggler wears a leather jacket. He carries a bottle of Altairan rum, some raspberries, and a laser pistol. The pistol is large. The jacket is large. The rum and the raspberries are contraband. The description of the smuggler is "He has a bestubbed chin and a sinister eye."

The tourist is a woman in the customs house. The description is "The type who walks off home with a dozen contraband items in her pocket not because she means to steal things but because she's too stupid to understand that the law applies to herself." She wears a tight-fitting dress. The dress is large. The tourist carries a grapevine and an archaeological artifact. The grapevine is large and contraband. The artifact is contraband.

Report examining someone: say "[The noun] is [if the noun is wearing something]wearing [a list of unconcealed things worn by the noun] and [end if]carrying [a list of unconcealed things carried by the noun]."

Report examining someone who is concealing something contraband:  
say "[The noun] looks nervous. You can just tell."

Report examining someone who is carrying an unconcealed contraband thing:  
say "Your eye goes at once to [the list of unconcealed contraband things carried by the noun] which [the noun] is unable to hide";



if an unconcealed licit thing is had by the noun, say ", though [if the noun is female]s[end if]he also has [a list of unconcealed licit things had by the noun]"; say "." instead.

Notice that we can talk about what the smuggler wears, what he carries, and what he "has": things the smuggler has can be either worn or carried, so the phrase is useful if we don't care to make that distinction.

Instead of searching someone:

say "[The noun] is revealed to be carrying [a list of things carried by the noun]."

Instead of confiscating the dress:

say "You are not allowed to perform strip-searches in the public customs area."

Understand "confiscate [something]" as confiscating.

Confiscating is an action applying to one thing.

Check confiscating:

unless the noun is had by someone who is not the player:

say "You can only confiscate other people's possessions." instead.

Carry out confiscating:

now the noun is carried by the player.

Report confiscating:

say "Through the authority vested in you by the power of the Sovereign of Centauri Proxima, you make [the noun] your own."

Test me with "x smuggler / search smuggler / x smuggler / confiscate jacket / x smuggler / confiscate rum / confiscate pistol / x smuggler / confiscate raspberries / x smuggler".

Test more with "x tourist / confiscate dress / confiscate grapevine / x tourist / search tourist / confiscate artifact / x tourist".

---

39



### Example Van Helsing

RB

A character who approaches the player, then follows him from room to room.

Suppose we want to write a character who tries to be in the same room as the player. We will do this by testing every turn whether the character's location and the player's location are the same; if the answer is no, the character will look for a path to the player's location, then try to move along that path. (We will learn more about finding paths and giving characters instructions later.)

The result will be that if the player ever moves to another location, the character will automatically pursue him.

"Van Helsing"

The Drawbridge is a room. North of the Drawbridge is the Immensely Enormous Entry Hall. West of the Entry Hall is the Vast Dining Area. North of the Vast Dining Area is the Colossal Kitchen. The Spooky Guano-filled Attic is above the Entry Hall.

Count Dracula is a man in the Attic.

In the following condition, we could also have written "if the location of Count Dracula is not the location", because "location" by itself is always understood to be the player's location. But it seemed better for clarity to write it this way:

Every turn:

- if the location of Count Dracula is not the location of the player:
  - let the way be the best route from the location of Count Dracula to the location of the player, using doors;
  - try Count Dracula going the way;
- otherwise:
  - say "'Muhahaha,' says Count Dracula."

Test me with "z / z / n / w / e / u / z / d".

---

40



### Example Prisoner's Dilemma

RB

A button that causes a previously non-existent exit to come into being.

We can change the directions in the map in mid-game, though in practice this is rarely necessary. But suppose we do not want a door or any sign of a door to exist before the player takes some action, in this case pressing a button:

"Prisoner's Dilemma"

Challenger's Waiting Room is a room. "The challenge is this: to wait as long as you can endure to do so in a room with no features and no clock. If you wait longer than all the other contestants, you win."

The button is fixed in place in the Challenger's Waiting Room. "The only item in view is a black recessed button."

Amid the Cheering Throng is a room.

Instead of pushing the button for the first time:

- change the east exit of the Challenger's Waiting Room to Amid the Cheering Throng;
- change the west exit of the Cheering Throng to the Challenger's Waiting Room;
- say "With a groan of gears, the east wall swings open! If you've lost now, well, you've lost..."

Test me with "e / push button / e / w".

Our instructions about pushing the button will be further explained in the chapter on Actions, but the thing to note here is that we can "change (whatever) exit" in order to set or re-set map directions. Notice that we have to set both directions explicitly: changing the east exit of the Waiting Room does not automatically also change the west exit of Amid the Cheering Throng.

This allows greater flexibility in our games but does require an extra line or so of work.

---

41



### Example The World of Charles S. Roberts

RB

Replacing the ordinary compass bearings with a set of six directions to impose a hexagonal rather than square grid on the landscape.

Wargaming is an ancient pursuit, but its modern form began as a professional training exercise in 19th-century Prussian staff colleges; since at least as early as H. G. Wells's "Little Wars" (1913) it has been a hobby of "boys from twelve years of age to one hundred and fifty and for that more intelligent sort of girl who likes boys' games and books." The free-form tabletop game used miniature figures and tape-measured movements, and remains the dominant form today. But in the mid-20th century, map grids on printed sheets gave the hobby a sudden new lease of life. They were easier to set up, more interesting to look at, cheaper to sell by mail-order. 1970s sales figures for "Strategy and Tactics", the leading US subscription-based wargame distributor, were very similar to those of Infocom's IF games in the 1980s. And like classical IF, the grid-based wargame parceled up a continuous world into locations.

Grids were initially square, as on a chessboard, but square cells have several disadvantages. Four directions of movement (N, E, S, W) is too few, yet allowing movement in the diagonal directions means allowing tanks to travel about 1.4 times faster northeast than they do north. Square grids also only conform cleanly to man-made landscape features such as buildings in one orientation, and they never fit hills well. (A compromise measure to fix this, cutting the squares into octagons to leave smaller diamond squares at corner intersections, has never caught on.)



But following Charles S. Roberts's American Civil War designs for Avalon Hill of 1958-61 (notably "Chancellorsville" and the second edition of "Gettysburg"), a hexagonal grid became the new standard. Each hexagon is the same distance from the centre of all six of its neighbours, which are at equal angular spacings; and clumps of hexagons fit the shape of lakes, contoured hills, and so forth, much more naturally than clumps of squares do. Hexes also have a certain mystique - an air of "I don't belong in the children's department".

But hexes are tricky for IF, not least because English lacks words for "the direction 60 degrees around from front". Our cognitive view of the world tends to be square, perhaps because our two eyes both face front, in a direction at right angles to the plane of our arms, legs, pelvis and eyes. We reach out sideways at right angles to our walking. Even early hex-grid wargames called the cells "squares", though "hexes" eventually caught on. Still and all:

#### "The World of Charles S. Roberts"

Forward is a direction. Forward has opposite backward. Understand "f" as forward.

Backward is a direction. Backward has opposite forward. Understand "b" and "back" as backward.

Forward left is a direction. Forward left has opposite backward right. Understand "fl" as forward left.

Forward right is a direction. Forward right has opposite backward left. Understand "fr" as forward right.

Backward left is a direction. Backward left has opposite forward right. Understand "bl" as backward left.

Backward right is a direction. Backward right has opposite forward left. Understand "br" as backward right.

Now to forbid the use of the compass directions:

A direction can be hexagonal or squared-off. A direction is usually squared-off. Forward, backward, forward left, forward right, backward left and backward right are hexagonal.

Before going a squared-off direction, say "In this hexagonally-divided landscape, squared-off directions are not allowed." instead.

A slight nuisance is that, with things as they are above, typing BACKWARD produces the response "Which do you mean, backward, backward left or backward right?" To avoid that silly question, we write:

Does the player mean going backward: it is very likely. Does the player mean going forward: it is very likely.

And now a clump of 37 hexes, in six columns of six or seven rooms each. There are many ingenious ways we could put this map together automatically, but instead we will take a deep breath and write:

E1 is forward of E2. "Open farmland." E2 is forward of E3. "The edge of woods." E3 is forward of E4. "Deep woodland." E4 is forward of E5. "Deep woodland." E5 is forward of E6. "The rear edge of woods." E6 is forward of E7. "The start of a road leading forward right." E7 is a room. "Grassland."

F1 is forward of F2. "The edge of farmland." F2 is forward of F3. "The edge of woods." F3 is forward of F4. "Clearing in woods." F4 is forward of F5. "Deep woodland." F5 is forward of F6. "A road runs backward left to forward right." F6 is a room. "The edge of grassland."

G1 is forward of G2. "Grassland." G2 is forward of G3. "The edge of farmland." G3 is forward of G4. "A copse of trees." G4 is forward of G5. "The backward edge of woodland." G5 is forward of G6. "A bend in the road, from backward left to backward right." G6 is forward of G7. "Open farmland." G7 is a room. "Open farmland."

H1 is forward of H2. "Grassland, bordered by a hedge to the right." H2 is forward of H3. "The edge of farmland, with a hedge to forward right." H3 is forward of H4. "A copse of trees." H4 is forward of H5. "Open farmland." H5 is forward of H6. "A passing place on the road, which bends forward left to forward right." H6 is a room. "Open farmland."

I1 is forward of I2. "The end of a forward road, blocked by hedges on all sides except backward." I2 is forward of I3. "A straight road runs forward to backward, with long hedges to left and right." I3 is forward of I4. "A straight road runs forward to backward, alongside a long hedge to right." I4 is forward of I5. "A straight road runs forward to backward, alongside a long hedge to right." I5 is forward of I6. "Where three roads, forward, backward left and backward right, meet. Forward right is a thick hedge." I6 is forward of I7. "Open farmland." I7 is a room. "Open farmland."

J1 is forward of J2. "Dense woodland, with a hedge to left." J2 is forward of J3. "Grassland, with a hedge to left." J3 is forward of J4. "The edge of farmland, with a hedge to left." J4 is a room. "Open farmland, with a long hedge blocking movement forward left, backward left or backward." J5 is forward of J6. "A road

running forward left to backward right, alongside a hedge." J6 is a room. "Open farmland."

F1 is forward right of E2 and backward right of E1. F2 is forward right of E3 and backward right of E2. F3 is forward right of E4 and backward right of E3. F4 is forward right of E5 and backward right of E4. F5 is forward right of E6 and backward right of E5. F6 is forward right of E7 and backward right of E6.

G1 is forward right of F1. G2 is forward right of F2 and backward right of F1. G3 is forward right of F3 and backward right of F2. G4 is forward right of F4 and backward right of F3. G5 is forward right of F5 and backward right of F4. G6 is forward right of F6 and backward right of F5.

H1 is forward right of G2 and backward right of G1. H2 is forward right of G3 and backward right of G2. H3 is forward right of G4 and backward right of G3. H4 is forward right of G5 and backward right of G4. H5 is forward right of G6 and backward right of G5. H6 is forward right of G7 and backward right of G6.

I3 is forward right of H3 and backward right of H2. I4 is forward right of H4 and backward right of H3. I5 is forward right of H5 and backward right of H4. I6 is forward right of H6 and backward right of H5.

J5 is forward right of I6 and backward right of I5. J6 is forward right of I7 and backward right of I6.

And now we have a hexagonally-gridded world. Route-finding will work; prepositional forms like "to be mapped backward left of" exist, just as they should; and in general these directions are just as good as the square ones. (The only thing which doesn't look good is the Index map, where Inform is just unable to draw a picture because it assumes a square grid. But that has no effect on play.)

The landscape is much easier to navigate with a little diagram:

```
To say legend (D - direction):
  let destination hex be the room D from the location;
  if the destination hex is nothing, say " ";
  otherwise say the destination hex.
```

```
Carry out looking:
  say "[fixed letter spacing] \ [legend forward] /[line break][legend forward left] ---
- [legend forward right][line break] / \[line break]--< [location] >--[line break] \
/[line break][legend backward left] ---- [legend backward right][line break] /
[legend backward] \[variable letter spacing][line break]".
```

And finally:

The player is in I5.

Test me with "f / forward / backward left / bl / br / br / f".



Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel.

Suppose we want to understand shipboard directions, but only when the player is aboard a vessel.

"Fore"

#### Section 1 - Procedure

The starboard is a direction. The starboard has opposite port. Understand "s" as starboard when the location is nautical.

The port is a direction. The port has opposite starboard. Understand "p" as port when the location is nautical.

The fore is a direction. The fore has opposite aft. Understand "f" as fore when the location is nautical.

The aft is a direction. The aft has opposite fore. Understand "a" as aft when the location is nautical.

Does the player mean going a nautical direction when the location is nautical: it is very likely.

Index map with fore mapped as north. Index map with aft mapped as south.  
Index map with port mapped as west. Index map with starboard mapped as east.

And we can even add new ways to talk about the ways things are mapped, borrowing from the Relations chapter. The following will allow us to use "is abaft of" as well as "is aft of":

[The verb to be abaft of means the mapping aft relation.]

Now, to prevent the player from using NORTH onboard ship, or AFT on land:

A room can be nautical or earthbound. A room is usually not nautical. A direction can be nautical or earthbound. A direction is usually not nautical. Starboard, port, fore, aft, up, down, the inside and the outside are nautical.

Before going a nautical direction when the location is not nautical, say "Nautical directions can only be used on board ship."

Before going an earthbound direction when the location is nautical, say "Compass directions make no sense on board ship, but you can use [list of nautical directions] instead." instead.

#### Section 2 - Scenario

The Fish Room is aft of the Spirit Room. Starboard of the Fish Room is the After Powder Magazine. The Bread Room is aft of the After Powder Magazine.



The Fish Room, the Spirit Room, the Bread Room, and the After Powder Magazine are nautical.

The description of the Fish Room is "Absurd quantities of salt fish are kept here, and periodically visited by the cook or someone serving him. It is otherwise an unexceptional little chamber, so far below the waterline that there are no portholes and no external light of any kind. [paragraph break]A narrow doorway leads forward into the Spirit Room, and the After Powder Magazine is starboard."

The description of the Spirit Room is "Despite its ghostly name, this is little more than a closet down at the very navel of the ship, in which alcohol is kept: both for purifying wounds and for drinking. Under normal circumstances there is a guard posted here at every hour, lest anyone take to raiding the larder. The current absence of the guard marine strikes you as a very bad sign indeed. [paragraph break]The only way out is aft."

The description of the Bread Room is "The Bread Room is not only tiny from side to side and front to back: it is also about half the height of a proper room, and the floor slopes up very steeply with the curve of the hull. [paragraph break]What is kept here would not, on land, be dignified by the name of bread: it's hard tack, punishing to the teeth, dry on the tongue, and usually a home to weevils before half the journey is done. [paragraph break]More headroom, and access to the rest of the ship, lies fore through the After Powder Magazine."

The description of the After Powder Magazine is "Kept in near darkness because no one with any sense would bring a naked flame down here: when necessary, it can be lit with a single small lantern made of very thick glass and sealed to keep the sparks within. Sacks of powder are passed up into the higher levels of the ship by the scrubby little boys called 'powder monkeys' -- but none such are here now."

Test me with "north / aft / fore".

---